Welcome to the Blue Teaming Free Training

Modules

- **Module 1** - Incident Response and Security Operations Fundamentals
- **Module 2** - TOP 20 Open-source tools every Blue Teamer should have
- **Module 3** - How to deploy your Elastic Stack (ELK) SIEM
- **Module 4** - Getting started using Microsoft Azure Sentinel (Cloud-Native SIEM and SOAR)
- **Module 5** - Hands-on Wazuh Host-based Intrusion Detection System (HIDS) Deployment
- **Module 6 -** Threat Intelligence Fundamentals:
- **Module 7 -** How to Install and use The Hive Project in Incident Management
- **Module 8 -** Incident Response and Threat hunting with OSQuery and Kolide Fleet
- **Module 9 -** How to use the MITRE PRE-ATT&CK framework to enhance your reconnaissance assessments
- **Module 10 -** How to Perform Open Source Intelligence (OSINT) with SpiderFoot
- **Module 11 -** How to perform OSINT with Shodan
- **Module 12 -** Using MITRE ATT&CK to defend against Advanced Persistent Threats
- **Module 13 -** Hands-on Malicious Traffic Analysis with Wireshark
- **Module 14 -** Digital Forensics Fundamentals
- **Module 15 -** How to Perform Static Malware Analysis with Radare2
- **Module 16 -** How to use Yara rules to detect malware
- **Module 17 -** Getting started with IDA Pro

- **Module 18 -** Getting Started with Reverse Engineering using Ghidra
- **Module 19 -** How to Perform Memory Analysis
- **Module 20 -** Red Teaming Attack Simulation with "Atomic Red Team"
- **Module 21 -** How to build a Machine Learning Intrusion Detection system
- **Module 22 -** Azure Sentinel - Process Hollowing (T1055.012) Analysis
- **Module 23 -** Azure Sentinel - Send Events with Filebeat and Logstash
- **Module 24 -** Azure Sentinel - Using Custom Logs and DNSTwist to Monitor Malicious Similar Domains

Code Snippets and Projects

- Azure Sentinel Code snippets and Projects

This training is maintained by: **Chiheb Chebbi**

> If you want me to modify/correct something please don't hesitate to contact me via: **chiheb-chebbi [at] outlook.fr**

Incident Response and Security Operations Fundamentals

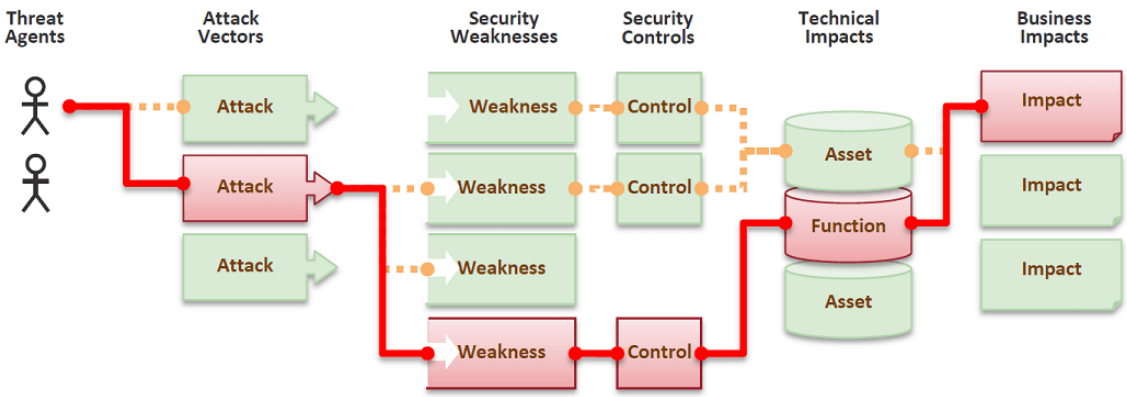
In this module, we are going to discover the required terminologies and fundamentals to acquire a fair understanding of “Incident Response” and the different steps and teams to perform incident response

We are going to explore the following points:

- Attack Vector Analysis
- Incident Response Fundamentals
- Incident Response Standards and Guidelines
- Incident response Process
- Incident response Teams
- Security Operation Centers

Before exploring what incident response is, let’s explore some important terminologies

Attack vector analysis Attack vectors are the paths used by attackers to access a vulnerability. In other words, the method used to attack an asset is called a Threat Vector or Attack vector. Attack vectors can be analyzed. The analysis is done by studying the attack surfaces like the entry points of an application, APIs, files, databases, user interfaces and so on. When you face a huge number of entries you can divide the modeling into different categories (APIs, Business workflows etc...)



Incident Response Fundamentals

TechTarget defines incident response as follows: “Incident response is an organized approach to addressing and managing the aftermath of a security breach or cyberattack, also known as an IT incident, computer incident or security incident. The goal is to handle the situation in a way that limits damage and reduces recovery time and costs.”

But what is an information security Incident?

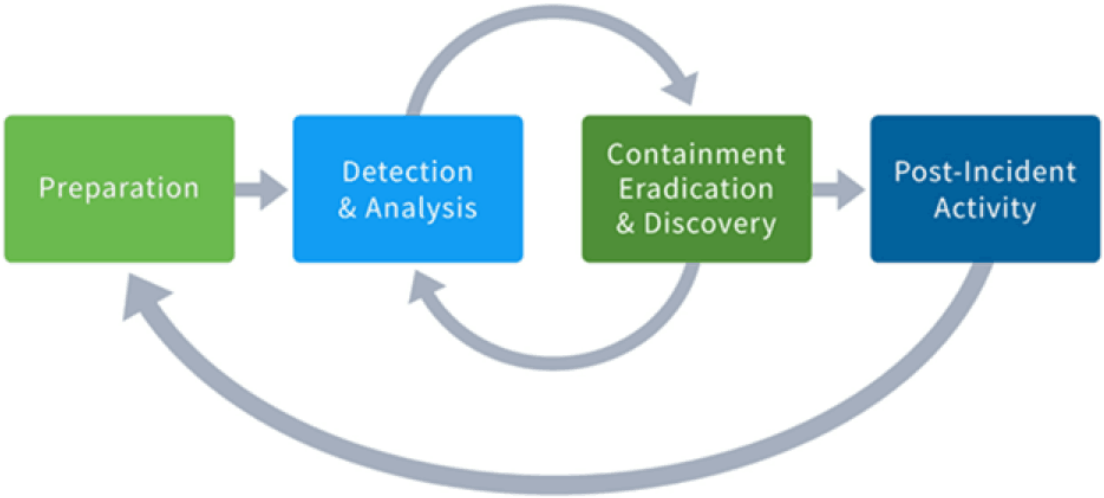
An event is any observable occurrence in a system or network. Events include a user connecting to a file share, a server receiving a request for a web page, a user sending email, and a firewall blocking a connection attempt. Incidents are events with a negative consequence, such as system crashes, packet floods, unauthorized use of system privileges, unauthorized access to sensitive data, and execution of malware that destroys data. During incident response operation there are a lot of artifacts resources you need to collect. You can use different artifacts such as:

- IP addresses
- Domain names
- URLs
- System calls
- Processes
- Services and ports
- File hashes

Incident Response Process

Incident response like any methodological operation goes thru a well-defined number of steps:

1. Preparation: during this phase, the teams deploy the required tools and resources to successfully handle the incidents including developing awareness training.
2. Detection and analysis: this is the most difficult phase. It is a challenging step for every incident response team. This phase includes networks and systems profiling, log retention policy, signs of an incident recognition and prioritizing security incidents.
3. Containment eradication and recovery: during this phase, the evidence pieces are collected and the containment and recovery strategies are maintained.
4. Post-incident activity: discussions are held during this phase to evaluate the team performance, to determine what actually happened, policies compliance and so on.



Establishing incident response teams

There are different incident response Teams: * Computer Security Incident Response Teams * Product Security Incident Response Teams * National CSIRTs and Computer Emergency Response Team.

Incident response standards and guidelines:

There are many great standards and guidelines to help you become more resilient and help you to build a mature incident response program some of the following: * Computer Security Incident Handling Guide: (NIST 800-63 Second revision), you can find it here: Computer Security Incident Handling Guide - NIST Page
* ISO 27035: ISO/IEC 27035 Security incident management * SANS Incident Handler Handbook: Incident Handler's Handbook - SANS.org * CREST Cyber Security Incident Response Guide: Cyber Security Incident Response Guide - crest

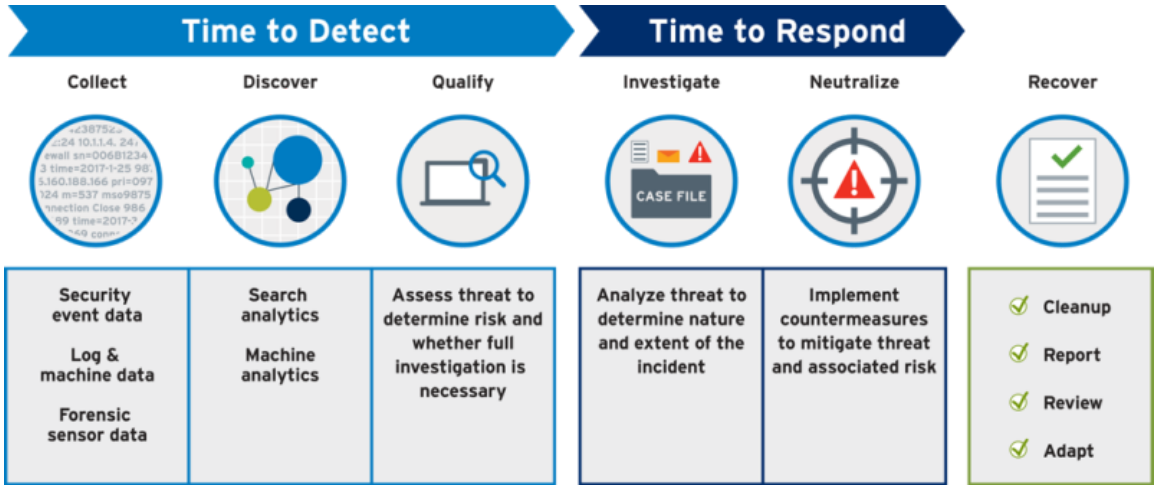
Security Operation Centers Fundamentals

Wikipedia defines Security Operation Centers as follows: A security operations center is a centralized unit that deals with security issues on an organizational and technical level. A SOC within a building or facility is a central location from where staff supervises the site, using data processing technology.

Security Operation Centers are not only a collection of technical tools. SOCs are people, process and technology.

To help you prepare your mission I highly recommend you to read this guide from Sampson Chandler : Incident Response Guide

It is essential to evaluate your SOC maturity because you can’t improve what you cannot measure. There are many maturity models in the wild based on different metrics based on your business needs and use cases. Some of the metrics are: * Time to Detect (TTD) * Time to Respond (TDR)



Your maturity model will be identified using this graph from LogRythm:

Summary

By now I assume that we covered many important terminologies and steps to perform incident response. The major goal of writing this article is delivering a collaborated guide to help our readers learning the fundamental skills needed in a daily basis job as incident handlers. Your comments are playing a huge role in this article. Please if you want to add or correct something please don't hesitate to comment so we can create together a one-stop resource for readers who are looking for a guide to learn about Incident Response. All your comments are welcome!

References and Credit

1. https://searchsecurity.techtarget.com/definition/incident-response
2. https://logrhythm.com/blog/a-ctos-take-on-the-security-operations-maturity-model/

TOP 20 Open-source tools every Blue Teamer should have

In this module we are going to explore the TOP 20 open source tools that every blue teamer should have:

The Hive

TheHive is a scalable 4-in-1 open source and free security incident response platform designed to make life easier for SOCs, CSIRTs, CERTs and any information security practitioner dealing with security incidents that need to be investigated and acted upon swiftly. Thanks to Cortex, our powerful free and open-source analysis engine, you can analyze (and triage) observables at scale using more than 100 analyzers.

Its official website: https://thehive-project.org

OSSIM

OSSIM is an open-source security information and event management system (SIEM). It was developed in 2003. The project was acquired later by AT&T.

You can download it from here: https://cybersecurity.att.com/products/ossim

The HELK

If you are into threat hunting than you probabilly heard of the HELK project. The HELK was developed by Roberto Rodriguez (Cyb3rWard0g) under GPL v3 License. The project was build based on the ELK stack in addition to other helpful tools like Spark, Kafka and so on.

Its official website: Cyb3rWard0g/HELK: The Hunting ELK - GitHub

Nmap

Scanning is one of the required steps in every attacking operation. After gathering information about a target you need to move on to another step which is scanning. If you are into information security you should have Nmap in your arsenal. Nmap (The abbreviation of Network mapper) is the most powerful network scanner. It is free and open-source. It gives you the ability to perform different types of network scans in addition to other capabilities thanks to its provided scripts. Also, you can write your own NSE scripts.

You can download it from here: https://nmap.org/download.html

Volatility

Memory malware analysis is widely used for digital investigation and malware analysis. It refers to the act of analyzing a dumped memory image from a targeted machine after executing the malware to obtain multiple numbers of artifacts including network information, running processes, API hooks, kernel loaded modules, Bash history, etc. Volatility is the most suitable tool to do that. It is an open-source project developed by volatility foundation. It can be run on Windows,Linux and MacOS. Volatility supports different memory dump formats including dd, Lime format, EWF and many other files.

You can download Volatility from here: https://github.com/volatilityfoundation/volatility

Demisto Community Edition

Security Orchestration, Automation and Response or simply SOAR are very effective platforms and tools to avoid analysts fatigue by automating many repetitive security tasks. One of the most-known platforms is Demisto. The platform provides also many free playbooks.

You can download the community edition from here: https://www.demisto.com/community/

Wireshark

Communication and networking are vital for every modern organization. Making sure that all the networks of the organization are secure is a key mission. The most suitable tool that will help you monitor your network is definitely Wireshark. Wireshark is a free and open-source tool to help you analyse network protocols with deep inspection capabilities. It gives you the ability to perform live packet capturing or offline analysis. It supports many operating systems including Windows, Linux, MacOS, FreeBSD and many more systems.

You can download it from here: https://www.wireshark.org/download.html

Atomic Red Team

Atomic __Red Team__ allows every __security team__ to test their controls by executing simple "atomic tests" that exercise the same __techniques__ used by adversaries (all mapped to Mitre's ATT&CK)

Its official website: https://github.com/redcanaryco/atomic-red-team

Caldera

Another threat simulation tool is Caldera.

CALDERA is an __automated__ adversary emulation system that performs post-compromise adversarial behavior within __WindowsEnterprise__ networks. It generates plans during operation using a planning system and a pre-configured adversary model based on the Adversarial Tactics, Techniques & Common Knowledge (ATT&CK™) project.

Its official website: https://github.com/mitre/caldera

Suricata

Intrusion detection systems are a set of devices or pieces of software that play a huge role in modern organizations to defend against intrusions and malicious activities. The role of network-based intrusion detection systems is to detect network anomalies by monitoring the inbound and outbound traffic. One of the most-used IDSs is Suricata. Suricata is an open-source IDS/IPS developed by the Open Information Security Foundation (OISF)

Its official website: https://suricata-ids.org

Zeek (Formely Bro IDS)

Zeek is one of the most popular and powerful NIDS. Zeek was known before by Bro. This network analysis platform is supported by a large community of experts. Thus, its documentation is very detailed and good.

Its official website: https://www.zeek.org

OSSEC

OSSEC is a powerful host-based intrusion detection system. It provides Log-based Intrusion Detection (LIDs), Rootkit and Malware Detection, Compliance Auditing, File Integrity Monitoring (FIM) and many other capabilities.

Its official website:https://www.ossec.net

OSQuery

OSQuery is a framework that is supported by many operating systems in order to perform system analytics and monitoring using simple queries. It uses SQL queries.

Its official website:https://www.osquery.io

AccessData FTK Imager

Forensics imaging is a very important task in digital forensics. Imaging is copying the data carefully with ensuring its integrity and without leaving out a file because it is very critical to protect the evidence and make sure that it is properly handled. That is why there is a difference between normal file copying and imaging. Imaging is capturing the entire drive. When imaging the drive, the analyst image the entire physical volume including the master boot record. One of the used tools is "AccessData FTK Imager".

Its official website: https://accessdata.com/product-download/ftk-imager-version-4-2-0

Cuckoo

Malware analysis is the art of determining the functionality, origin and potential impact of a given malware sample, such as a virus, worm, trojan horse, rootkit, or backdoor. As a malware analyst, our main role is to collect all the information about malicious software and have a good understanding of what happened to the infected machines. The most-known malware sandbox is cuckoo.

Its official website: https://cuckoo.sh/blog/

MISP

Malware Information Sharing Platform or simply MISP is an open-source threat sharing platform where analysts collaborate and share information about the latest threats between them. The project was developed by Christophe Vandeplas and it is under GPL v3 license.

Its official website:https://www.misp-project.org

Ghidra

Another great reverse engineering tool is Ghidra. This project is open-source and it is maintained by the National Security Agency Research Directorate. Ghidra gives you the ability to analyze different file formats. It supports Windows, Linux and MacOS. You need to install Java in order to run it. The project comes with many helpful detailed training, documentation and cheat-sheets. Also, it gives you the ability to develop your own plugins using Java or Python.

Its official website is: http://ghidra-sre.org

Snort

Another powerful network-based intrusion detection system is Snort. The project is very powerful and it was developed more than 5 million times. Thus, it is well documented and it is supported by a large community of network security experts.

Its official website: https://www.snort.org

Security Onion

If you are looking for a ready-to-use OS that contains many of the previously discussed tools you can simply download Security Onion. IT is a free and open-source Linux distribution for intrusion detection, enterprise security monitoring, and log management.

Its official website:https://github.com/Security-Onion-Solutions/security-onion

Detailed Guide: How to deploy your Elastic Stack (ELK) SIEM

Security information and event management systems (SIEM) are very important tools in incident response missions. Every security operation centre is equipped with a SIEM. In this article, we are going to learn how to deploy a fully working SIEM using the amazing suite the Elastic stack (ELK).

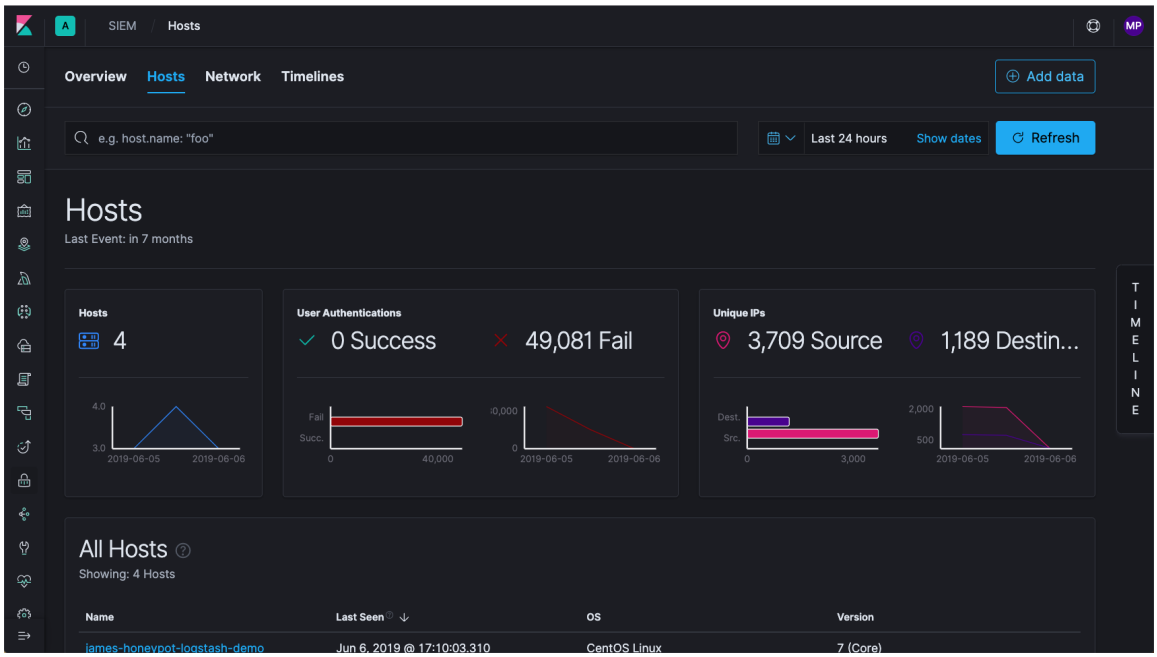


Image source: dashboard

In this article we are going to explore the following points:

- What is Elastic stack?
- How to install Elastic stack?
- How to install Elasticsearch?
- How to install kibana?
- How to install logstach?
- How to deploy ELK beats: **Metricbeat**
- How to deply **Auditbeat**
- How to deploy an **ELK SIEM**

Before diving deep into the required steps to build a SIEM, it is essential to acquire a fair understanding of the different ELK components.

What is the ELK Stack?

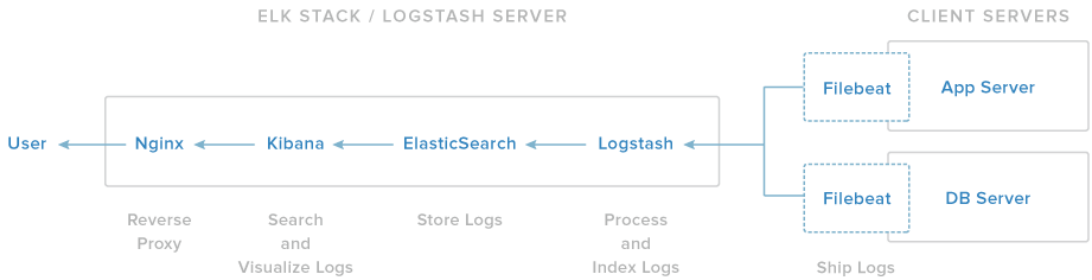


Image source: ELK

ELK Stack is the abbreviated form of "Elasticsearch Logstash Kibana" Stack. They are three open source projects. This stack is one of the world's most popular log management platforms by 500,000 downloads every month. The ELK stack is widely used in information technology businesses because it provides business intelligence, security and compliance, and web analytics.

Let's get started;

To build the SIEM, you need to install the required libraries and programs:

For the demonstration, I used a **Ubuntu 18.04** server hosted on Microsoft Azure

Update the sources.list file:

```
sudo apt update
```

```
Hit:1 http://azure.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://azure.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB
]
Get:4 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:5 http://azure.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages
[665 kB]
Get:6 http://azure.archive.ubuntu.com/ubuntu bionic-updates/main Translation-en
[246 kB]
Get:7 http://azure.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packa
ges [961 kB]
Get:8 http://azure.archive.ubuntu.com/ubuntu bionic-updates/universe Translation
-en [285 kB]
Get:9 http://azure.archive.ubuntu.com/ubuntu bionic-backports/universe amd64 Pac
kages [3736 B]
Get:10 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [43
6 kB]
Get:11 http://security.ubuntu.com/ubuntu bionic-security/main Translation-en [15
3 kB]
Get:12 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages
 [569 kB]
Get:13 http://security.ubuntu.com/ubuntu bionic-security/universe Translation-en
 [185 kB]
```

Install Java JDK 8 (and *apt-transport-https* if you are using Debian)

```
sudo apt install -y openjdk-8-jdk
```

```
azureuser@ELKSIEM:~$ sudo apt install -y openjdk-8-jdk wget apt-transport-https
Reading package lists... Done
Building dependency tree
Reading state information... Done
wget is already the newest version (1.19.4-1ubuntu2.2).
wget set to manually installed.
The following additional packages will be installed:
  adwaita-icon-theme at-spi2-core ca-certificates-java fontconfig
  fontconfig-config fonts-dejavu-core fonts-dejavu-extra gtk-update-icon-cache
  hicolor-icon-theme humanity-icon-theme java-common libasound2
  libasound2-data libasyncns0 libatk-bridge2.0-0 libatk-wrapper-java
  libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data libatspi2.0-0 libcairo2
  libcroco3 libdatrie1 libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2
  libdrm-radeon1 libflac8 libfontconfig1 libfontenc1 libgail-common libgail18
  libgdk-pixbuf2.0-0 libgdk-pixbuf2.0-bin libgdk-pixbuf2.0-common libgif7
  libgl1 libgl1-mesa-dri libgl1-mesa-glx libglapi-mesa libglvnd0 libglx-mesa0
  libglx0 libgraphite2-3 libgtk2.0-0 libgtk2.0-bin libgtk2.0-common
  libharfbuzz0b libice-dev libice6 libjbig0 libjpeg-turbo8 libjpeg8 liblcms2-2
  libllvm7 libnspr4 libnss3 libogg0 libpango-1.0-0 libpangocairo-1.0-0
  libpangoft2-1.0-0 libpciaccess0 libpcsclite1 libpixman-1-0
  libpthread-stubs0-dev libpulse0 librsvg2-2 librsvg2-common libsensors4
  libsm-dev libsm6 libsndfile1 libthai-data libthai0 libtiff5 libvorbis0a
  libvorbisenc2 libx11-dev libx11-doc libx11-xcb1 libxau-dev libxaw7
  libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-render0
```

Check the Java version with:

```
java -version
```

```
openjdk version "1.8.0_212"
OpenJDK Runtime Environment (build 1.8.0_212-8u212-b03-0ubuntu1.18.04.1-b03)
OpenJDK 64-Bit Server VM (build 25.212-b03, mixed mode)
```

Now let's install Elasticsearch:

`wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -`

`echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list`

`sudo apt update`

`sudo apt install elasticsearch`

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  elasticsearch
0 upgraded, 1 newly installed, 0 to remove and 30 not upgraded.
Need to get 337 MB of archives.
After this operation, 536 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/7.x/apt stable/main amd64 elasticsea
rch amd64 7.2.0 [337 MB]
Fetched 337 MB in 7s (47.9 MB/s)
```

After installing elasticsearch you need to configure it by modifying **/etc/elasticsearch/elasticsearch.yml** file

`sudo vi  /etc/elasticsearch/elasticsearch.yml`

```
#
#bootstrap.memory_lock: true
#
# Make sure that the heap size is set to about half the memory available
# on the system and that the owner of the process is allowed to use this
# limit.
#
# Elasticsearch performs poorly when the system is swapping the memory.
#
# ---------------------------------- Network -----------------------------------
#
# Set the bind address to a specific IP (IPv4 or IPv6):
#
network.host: 0.0.0.0
#
# Set a custom port for HTTP:
#
http.port: 9200
#
# For more information, consult the network module documentation.
#
# --------------------------------- Discovery ----------------------------------
#
```

Un-comment **network.host** and **http.port** and assign values to them. Don't use "0.0.0.0" in your production servers. I am using it just for a demonstration.

save the file.

To start Elasticsearch on boot up type:

```
sudo update-rc.d elasticsearch defaults 95 10
```

Start elasticsearch service:

```
sudo service elasticsearch start
```

Check the installation:

```
curl -X GET "YOU_IP:9200"
```

```
{
  "name" : "ELK",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "3H-H4Xw8Ska4N9LsiYCR9Q",
  "version" : {
    "number" : "7.2.0",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "508c38a",
    "build_date" : "2019-06-20T15:54:18.811730Z",
    "build_snapshot" : false,
    "lucene_version" : "8.0.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

Now let's install Kibana:


```
sudo apt install -y kibana
```

```
azureuser@ELK:~$ sudo apt install -y kibana
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  kibana
0 upgraded, 1 newly installed, 0 to remove and 30 not upgraded.
Need to get 218 MB of archives.
After this operation, 558 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/7.x/apt stable/main amd64 kibana amd
64 7.2.0 [218 MB]
Fetched 218 MB in 5s (48.3 MB/s)
```

And like what we did with elasticsearch we need to configure it too:

`sudo vi /etc/kibana/kibana.yml`

```
# The maximum payload size in bytes for incoming server requests.
#server.maxPayloadBytes: 1048576

# The Kibana server's name.  This is used for display purposes.
#server.name: "your-hostname"

# The URLs of the Elasticsearch instances to use for all your queries.
elasticsearch.hosts: ["0.0.0.0:9200"]

# When this setting's value is true Kibana uses the hostname specified in the server.host
# setting. When the value of this setting is false, Kibana uses the hostname of the host
# that connects to this Kibana instance.
#elasticsearch.preserveHost: true

# Kibana uses an index in Elasticsearch to store saved searches, visualizations and
# dashboards. Kibana creates a new index if the index doesn't already exist.
kibana.index: ".kibana"

# The default application to load.
#kibana.defaultAppId: "home"

# If your Elasticsearch is protected with basic authentication, these settings provide
-- INSERT --                                                    28,38         21%
```

Un-comment and modify the following values:

```
server.port: 5601
server.host: "YOUR-IP-HERE"
elasticsearch.url: "http://YOUR-IP-HERE:9200"
```

Save the file, and perform what we did previously

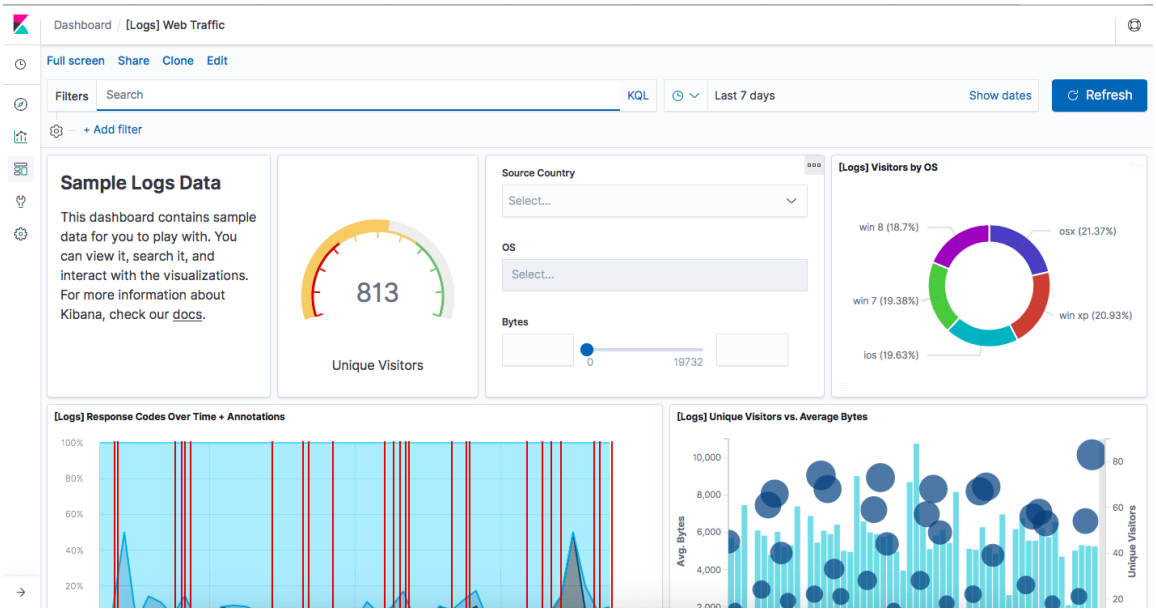
`sudo update-rc.d kibana defaults 95 10`

and run it:

`sudo service kibana start`

Now go to **https://YOUR-IP-HERE:5601**

Voila, you can start exploring the dashboard of some pre-installed Sample Log data:



Install **logstash** to collect, parse and transform logs if needed:


```
sudo apt install -y logstash
```

```
azureuser@ELK:~$ sudo apt install -y logstash
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  logstash
0 upgraded, 1 newly installed, 0 to remove and 30 not upgraded.
Need to get 173 MB of archives.
After this operation, 300 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/7.x/apt stable/main amd64 logstash a
ll 1:7.2.0-1 [173 MB]
Fetched 173 MB in 4s (46.4 MB/s)
Selecting previously unselected package logstash.
(Reading database ... 72500 files and directories currently installed.)
Preparing to unpack .../logstash_1%3a7.2.0-1_all.deb ...
Unpacking logstash (1:7.2.0-1) ...

Progress: [ 17%] [#########.....................................................]
```

But wait how can we use our own data?

It is a good question, we can receive data from a host using what we call "Beats". You can find the full list here:

As a demonstration i am going to use "Metricbeat

```
sudo apt-get install metricbeat
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  metricbeat
0 upgraded, 1 newly installed, 0 to remove and 30 not upgraded.
Need to get 37.6 MB of archives.
After this operation, 162 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/7.x/apt stable/main amd64 metricbeat amd64 7.2.0 [37.6 MB]
Fetched 37.6 MB in 1s (39.4 MB/s)
Selecting previously unselected package metricbeat.
(Reading database ... 171714 files and directories currently installed.)
Preparing to unpack .../metricbeat_7.2.0_amd64.deb ...
Unpacking metricbeat (7.2.0) ...
Setting up metricbeat (7.2.0) ...
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for systemd (237-3ubuntu10.22) ...
```

Configure the beat by typing

```
sudo vi /etc/metricbeat/metricbeat.yml
```

To start metricbeat on boot up type as usual

```
sudo update-rc.d metricbeat defaults 95 10
```

Start the beat:

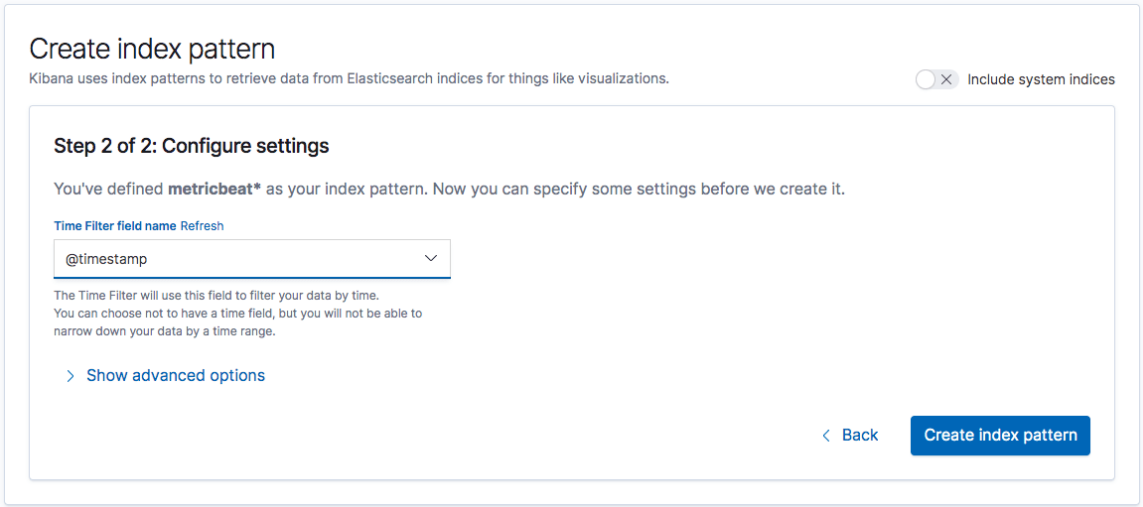
```
sudo service metricbeat start
```

Now go to the main dashboard and create a new index:

If everything went well you will see your beat:

metricbeat-7.2.0-2019.06.27-000001

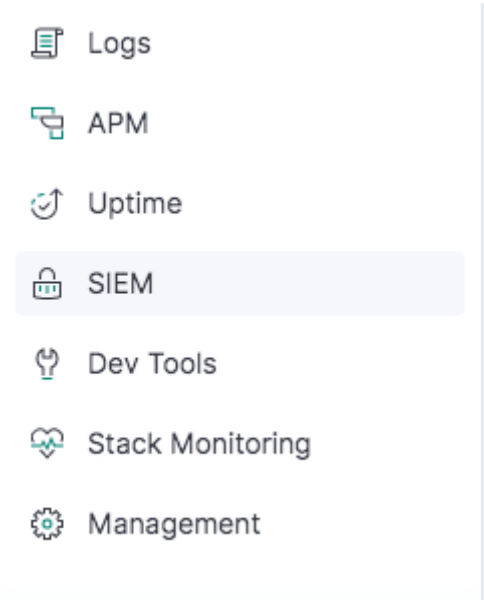
Select the time filter by selecting @timestamp:



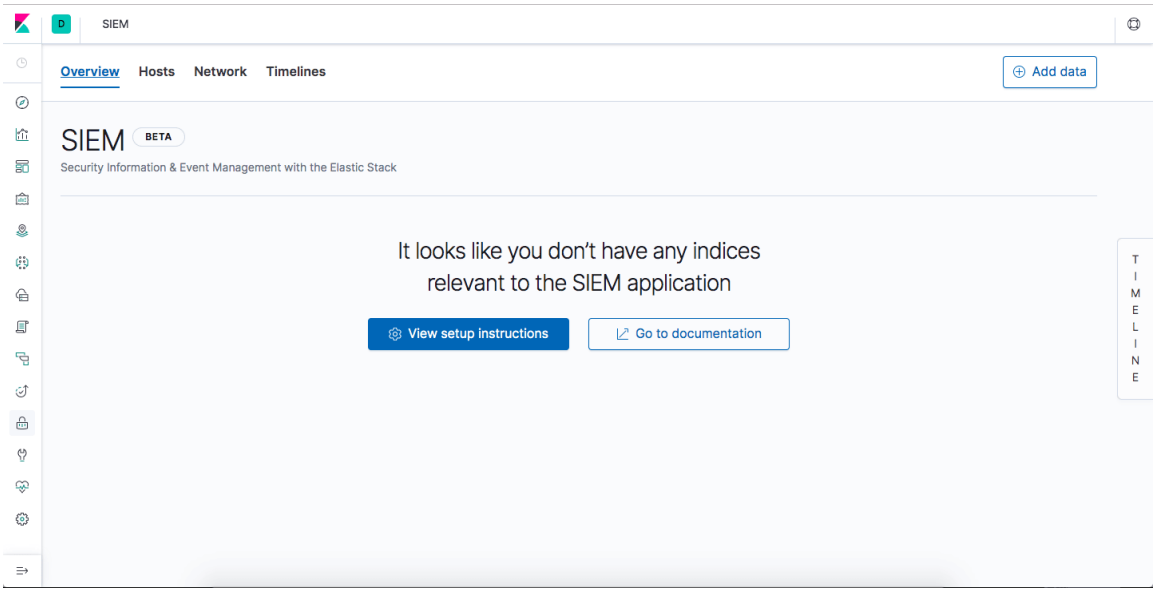
Then, you can visualize any data you want from that beat.

By now we deployed the most important parts. Let's learn how to deploy the ELK SIEM:

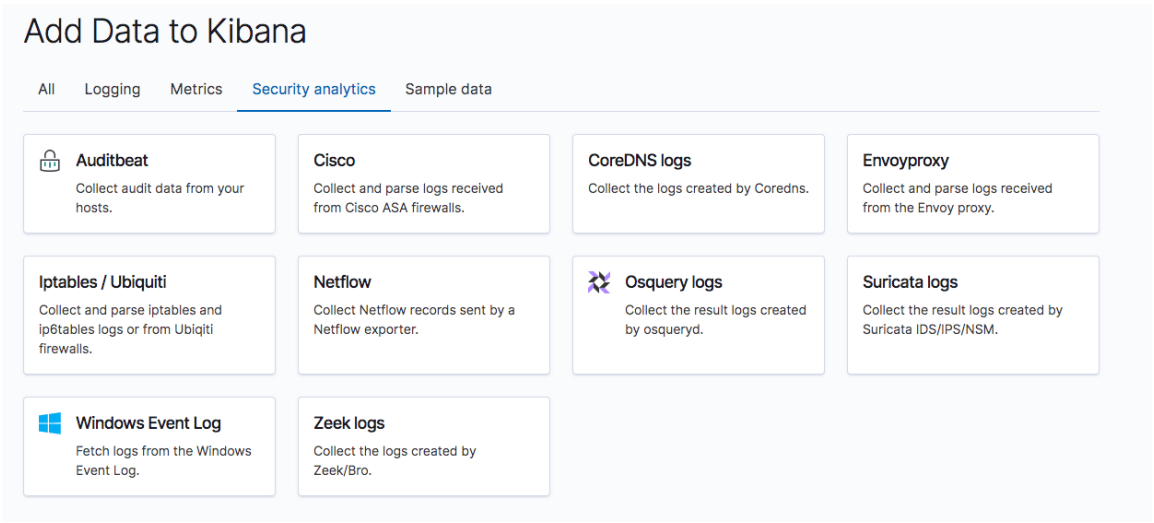
Go to the sidebar and you will find SIEM option:



It will take you to the main SIEM page:



But now we need data to run the SIEM. In order to do that we need to install other beats from sources like the following:



For the demonstration i am going to use the " **Auditbeat**":

```
sudo apt-get install auditbeat
```

Configure it by:

```
sudo vi /etc/auditbeat/auditbeat.yml
```

Check the setup:

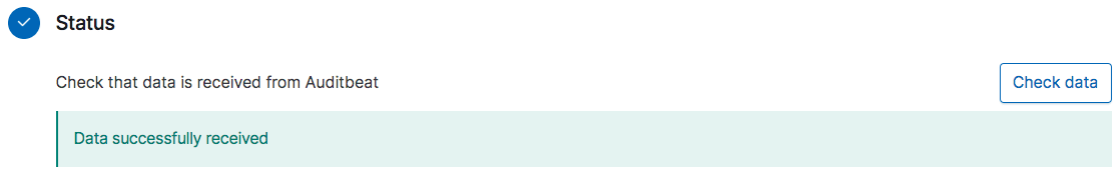
```
sudo auditbeat setup
```

```
Index setup finished.
Loading dashboards (Kibana must be running and reachable)
Loaded dashboards
```

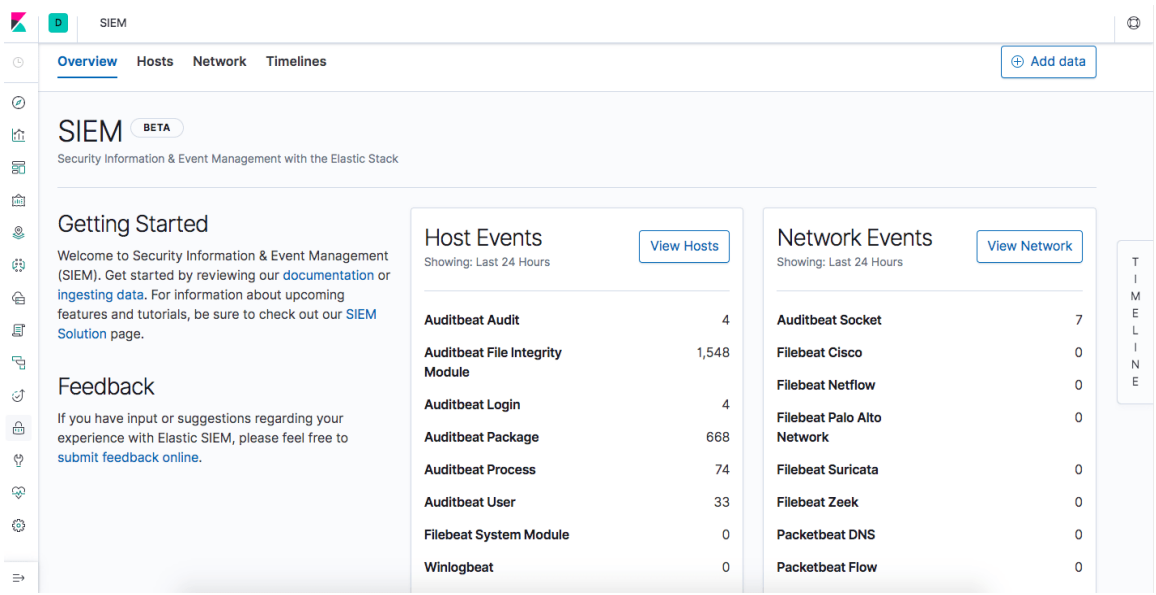
Run the beat:

```
sudo service auditbeat start
```

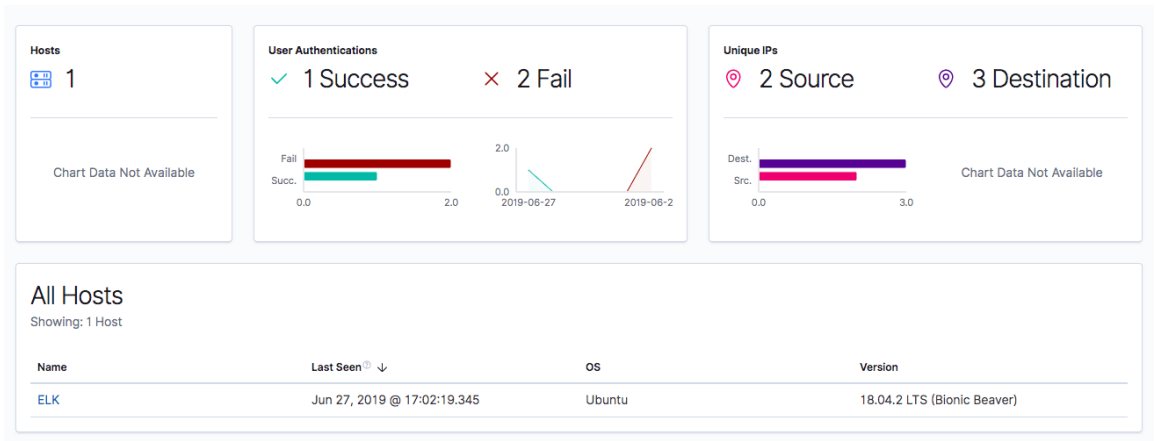
If you did everything correctly you will see this on the SIEM Dashboard:



Congratulations! Now you can see the dashboard of your SIEM.

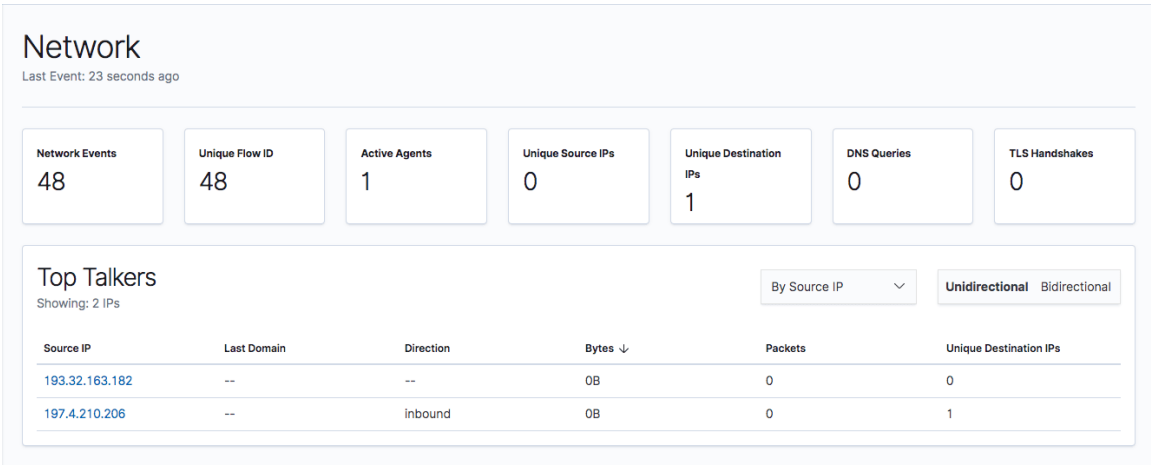


Check the hosts:

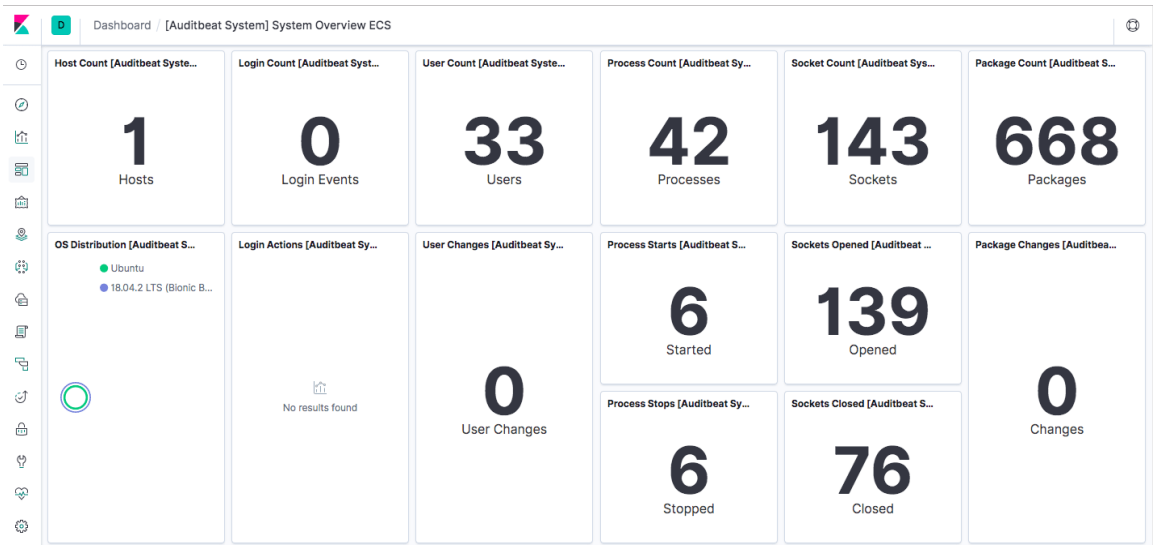


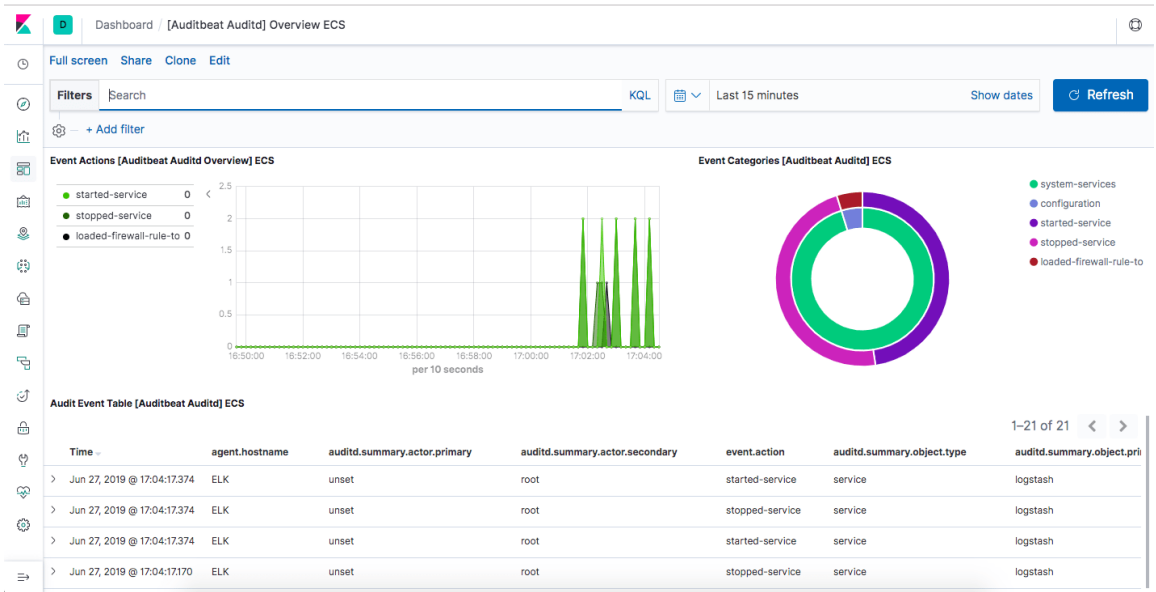
Timestamp	Host Name	Module/Dataset	Event Action	User	Source	Destination	Message
Jun 27, 2019 @ 17:02:19.345	ELK	system/socket	socket_opened	root	197.4.210.206:55270	10.0.4.6:5601	Inbound socket (197.4.210.206:55270 → 10.0.4.6:56 ▫▫▫
Jun 27, 2019 @ 17:02:19.345	ELK	system/socket	socket_opened	root	197.4.210.206:55278	10.0.4.6:5601	Inbound socket (197.4.210.206:55278 → 10.0.4.6:56 ▫▫▫
Jun 27, 2019 @ 17:02:19.345	ELK	system/socket	socket_opened	root	197.4.210.206:55266	10.0.4.6:5601	Inbound socket (197.4.210.206:55266 → 10.0.4.6:56 ▫▫▫
Jun 27, 2019 @ 17:02:19.345	ELK	system/socket	socket_opened	root	197.4.210.206:55249	10.0.4.6:5601	Inbound socket (197.4.210.206:55249 → 10.0.4.6:56 ▫▫▫
Jun 27, 2019 @ 17:02:19.345	ELK	system/socket	socket_opened	root	197.4.210.206:55275	10.0.4.6:5601	Inbound socket (197.4.210.206:55275 → 10.0.4.6:56 ▫▫▫
Jun 27, 2019 @ 17:02:19.345	ELK	system/socket	socket_opened	root	197.4.210.206:55247	10.0.4.6:5601	Inbound socket (197.4.210.206:55247 → 10.0.4.6:56 ▫▫▫
Jun 27, 2019 @ 17:02:19.345	ELK	system/socket	socket_opened	root	197.4.210.206:55258	10.0.4.6:5601	Inbound socket (197.4.210.206:55258 → 10.0.4.6:56 ▫▫▫
Jun 27, 2019 @ 17:02:19.345	ELK	system/socket	socket_opened	root	197.4.210.206:55251	10.0.4.6:5601	Inbound socket (197.4.210.206:55251 → 10.0.4.6:56 ▫▫▫
Jun 27, 2019 @ 17:02:19.345	ELK	system/socket	socket_opened	root	197.4.210.206:55243	10.0.4.6:5601	Inbound socket (197.4.210.206:55243 → 10.0.4.6:56 ▫▫▫

Check the Network Dashboard:



A system Overviews:

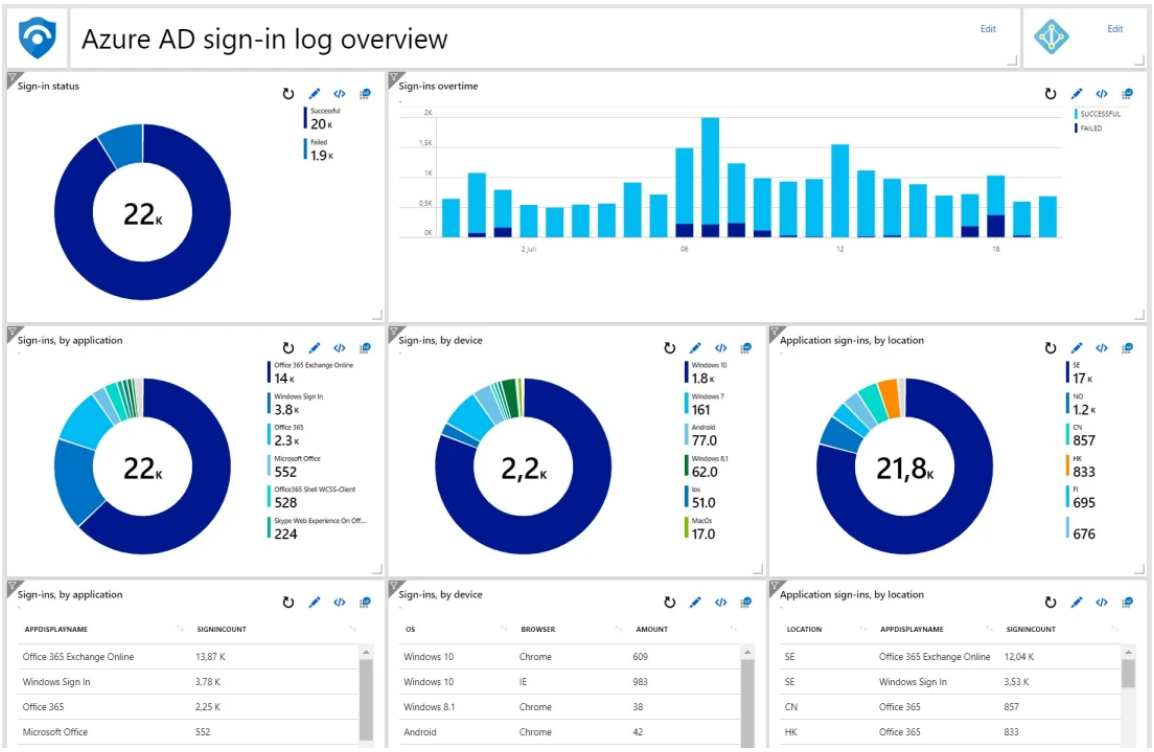




Voila, you learned how to build an ELK SIEM.

Getting started using Microsoft Azure Sentinel (Cloud-Native SIEM and SOAR)

In this module, we are going to explore Microsoft Azure Sentinel (Cloud-Native SIEM and SOAR). We are going to learn how to deploy the SIEM from scratch and we are going to see how to start detecting threats with it



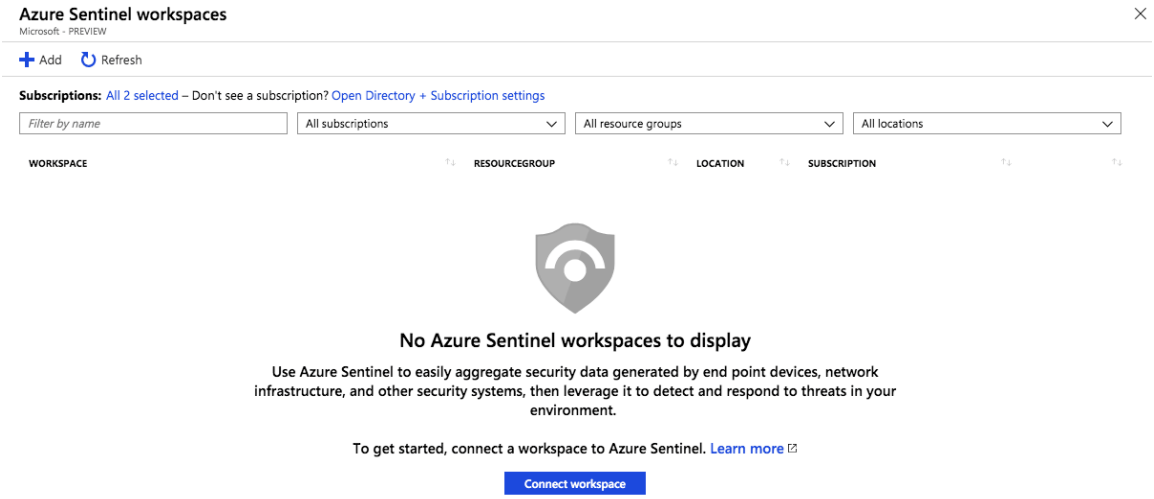
Source

Before learning how to use Azure Sentinel, we need to define it first. According to one of their official blog posts:

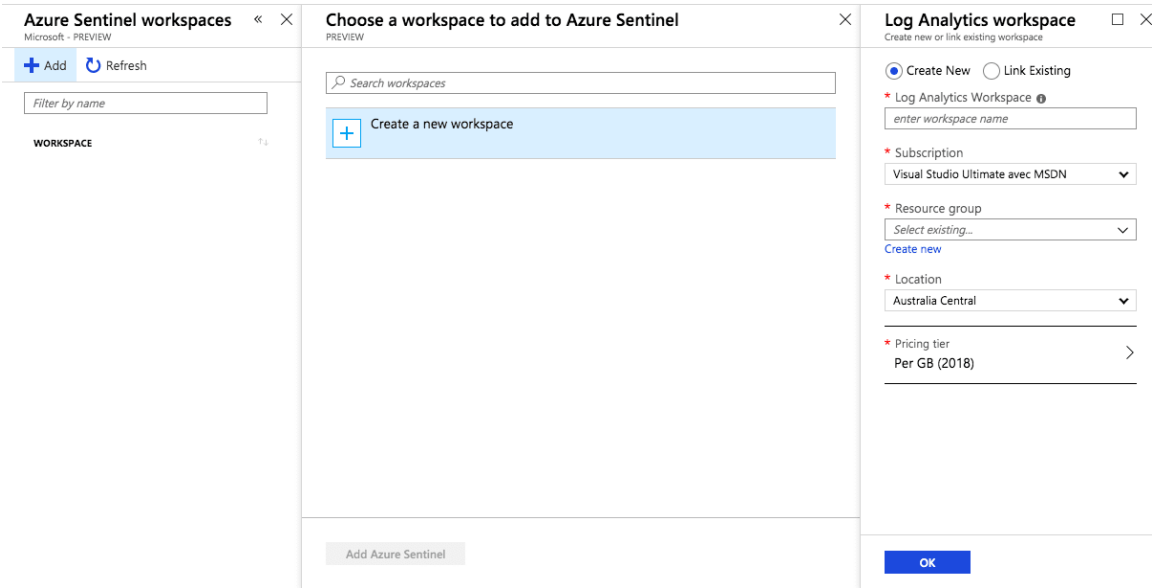
> Azure Sentinel provides intelligent security analytics at cloud scale for your entire enterprise. Azure Sentinel makes it easy to collect security data across your entire hybrid organization from devices, to users, to apps, to servers on any cloud. It uses the power of artificial intelligence to ensure you are identifying real threats quickly and unleashes you from the burden of traditional SIEMs by eliminating the need to spend time on setting up, maintaining, and scaling infrastructure.

Most of the first steps are already discussed in details in the previous resource. Thus I am going to go through the steps rapidly:

Go to Azure search bar and look for Azure Sentinel (preview) and add a new workplace

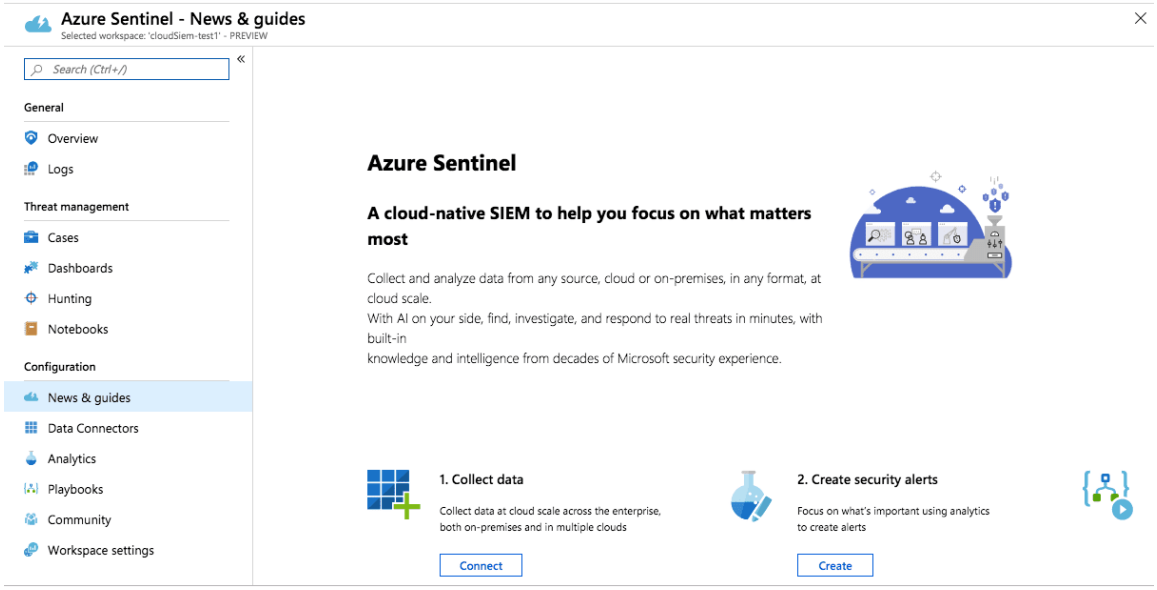


Create a new Workspace and press "OK"

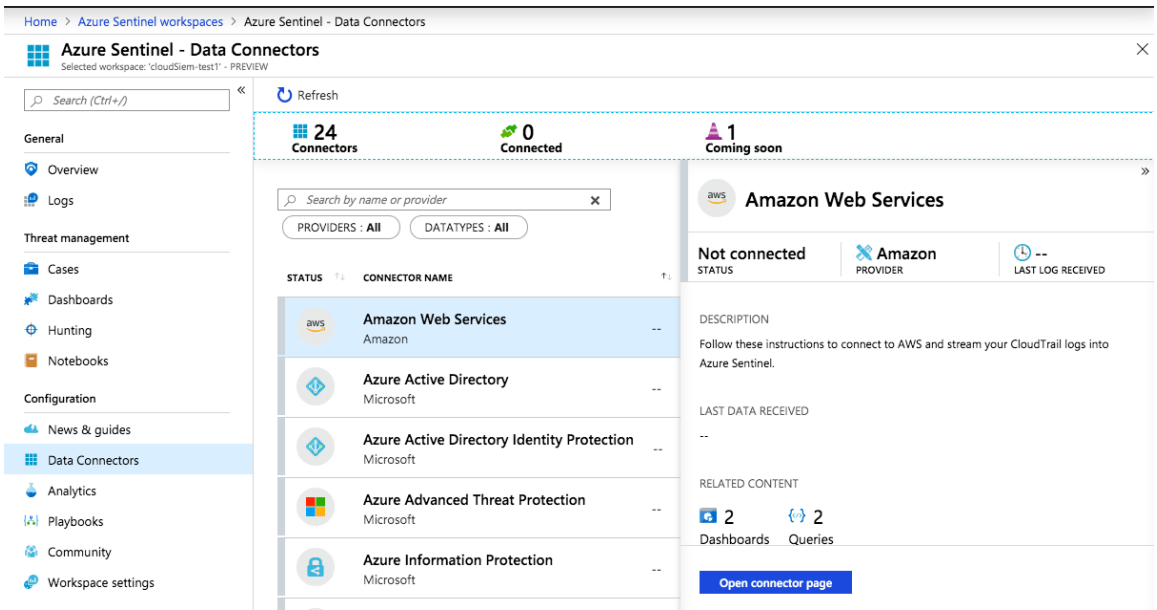


Add a new Azure Sentinel

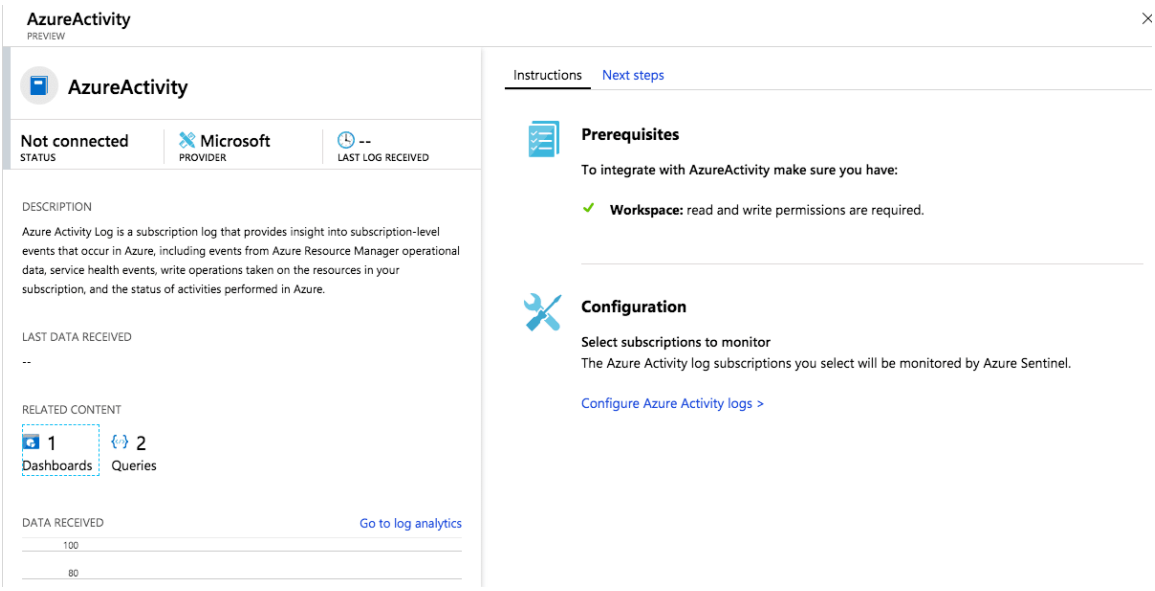
Voila!



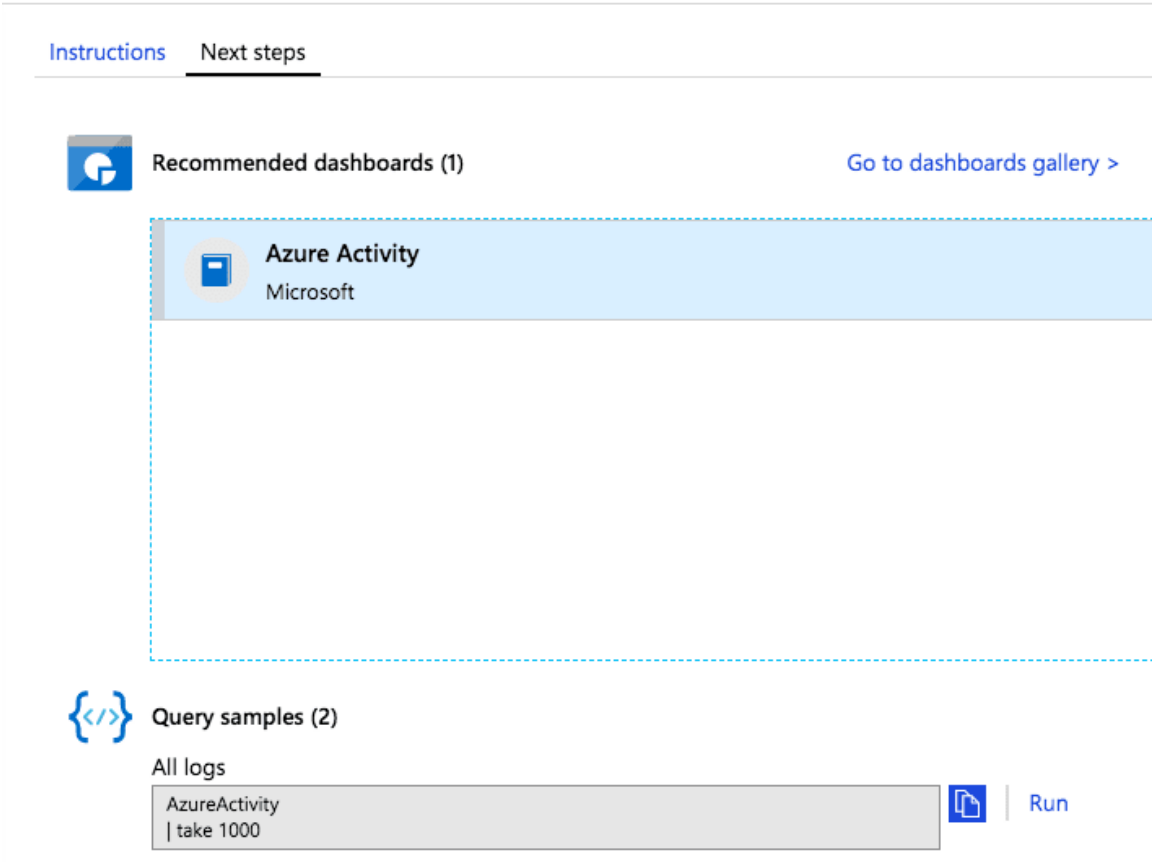
Now you need to select a connector to receive logs:

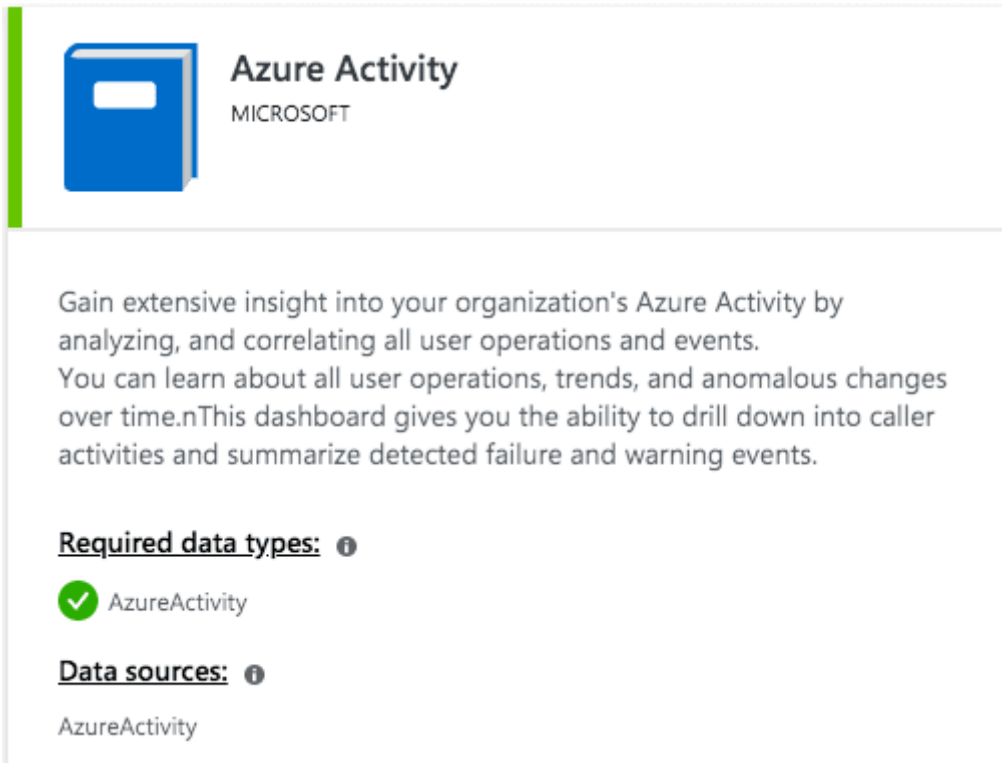


For example, you can select Azure Activities:

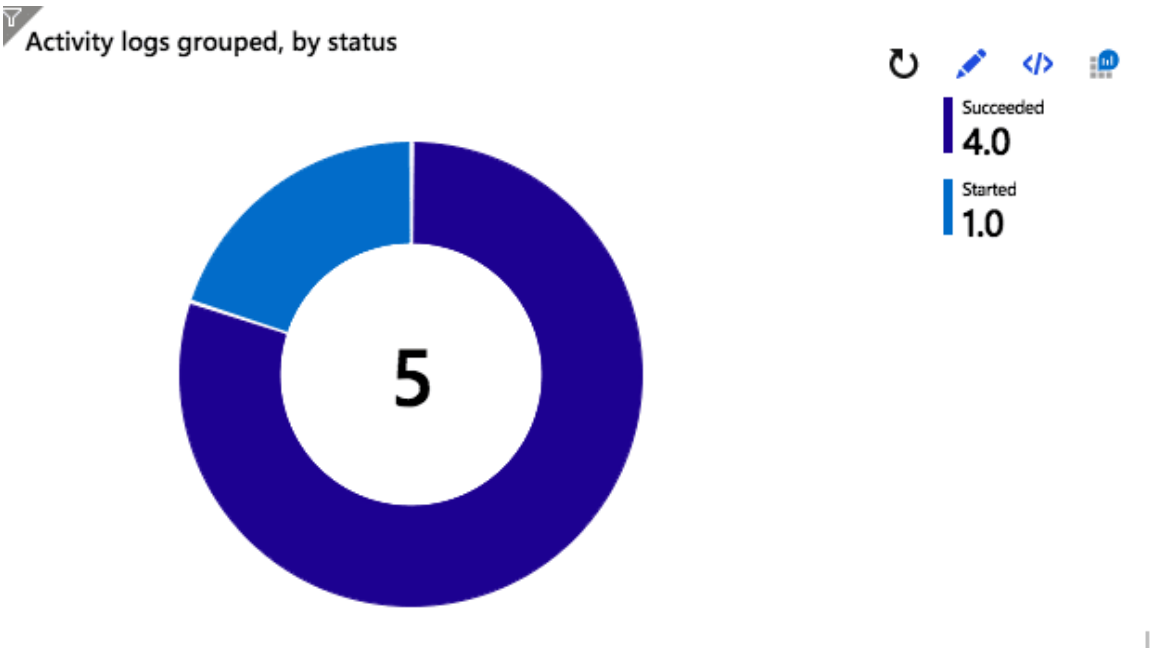


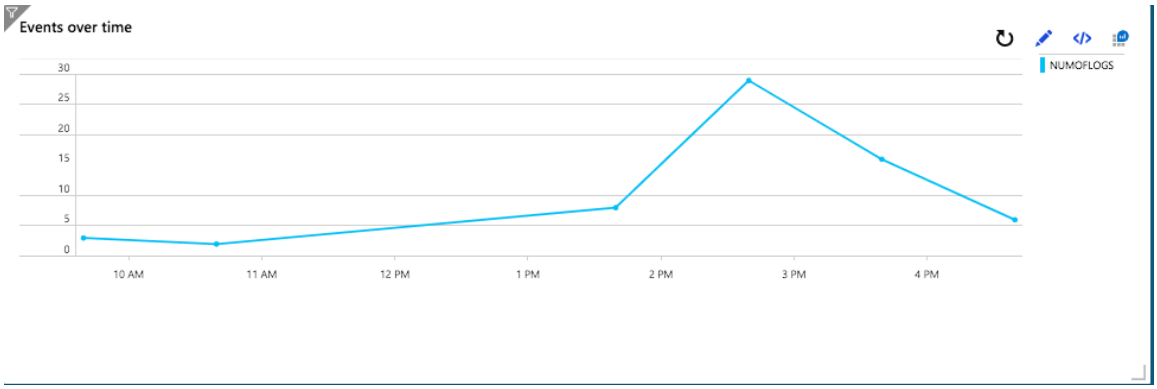
Click "Next Steps"

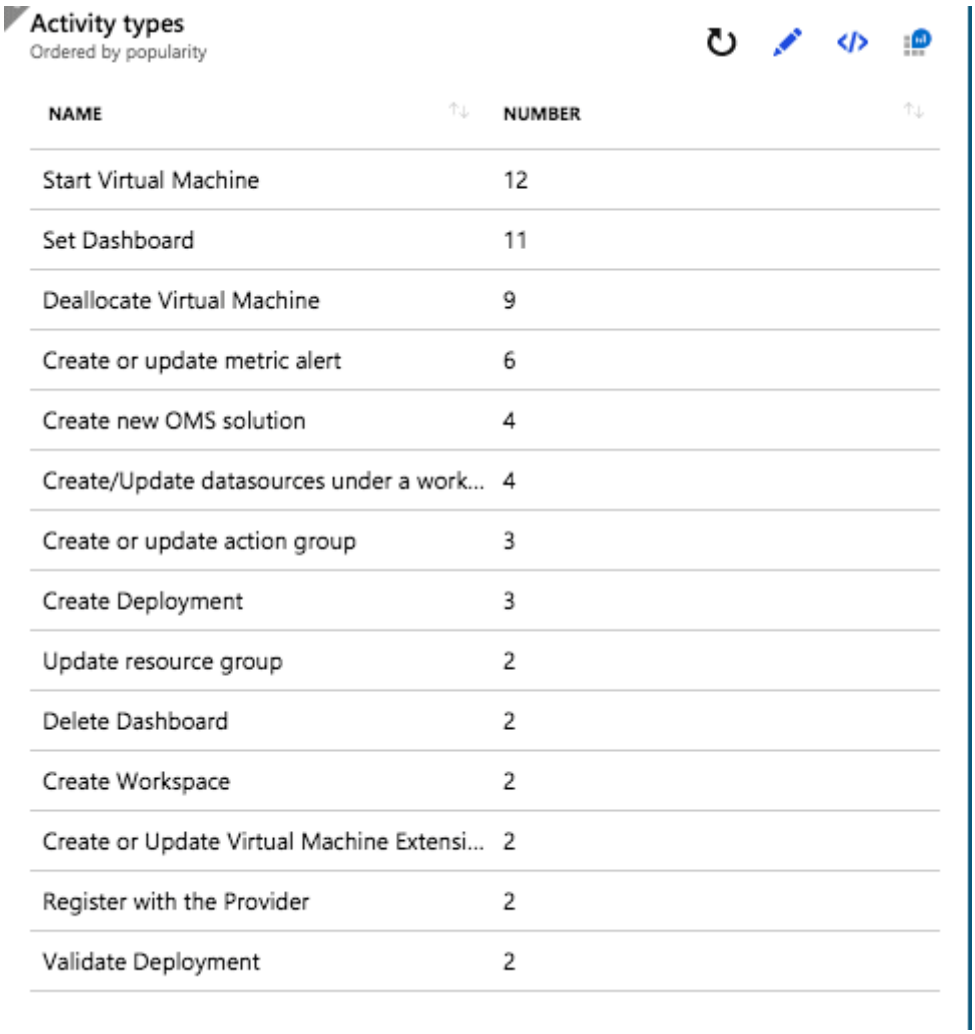




Create a Dashboard. The following graph illustrates some of the Dashboard components:

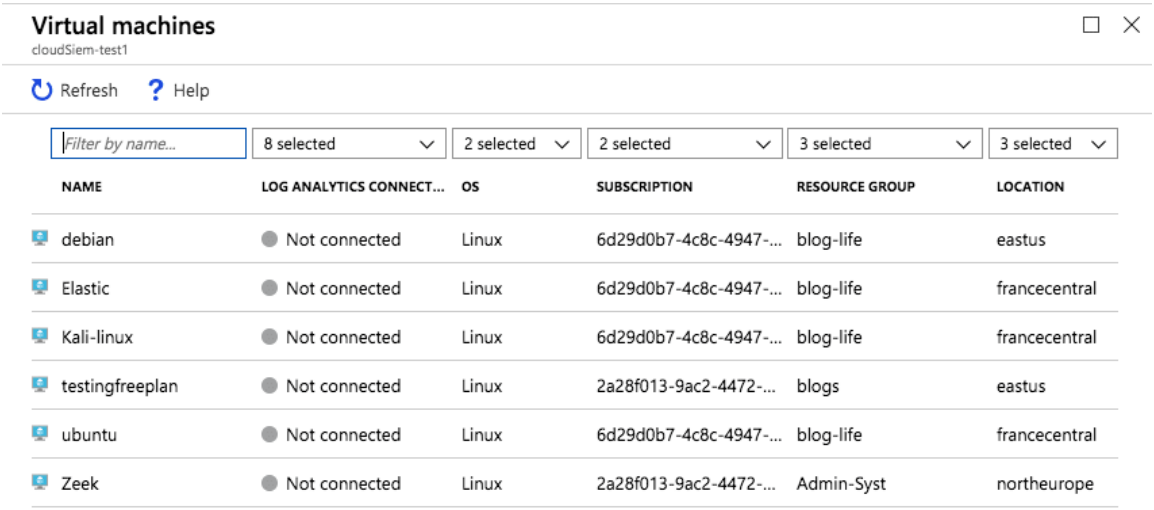




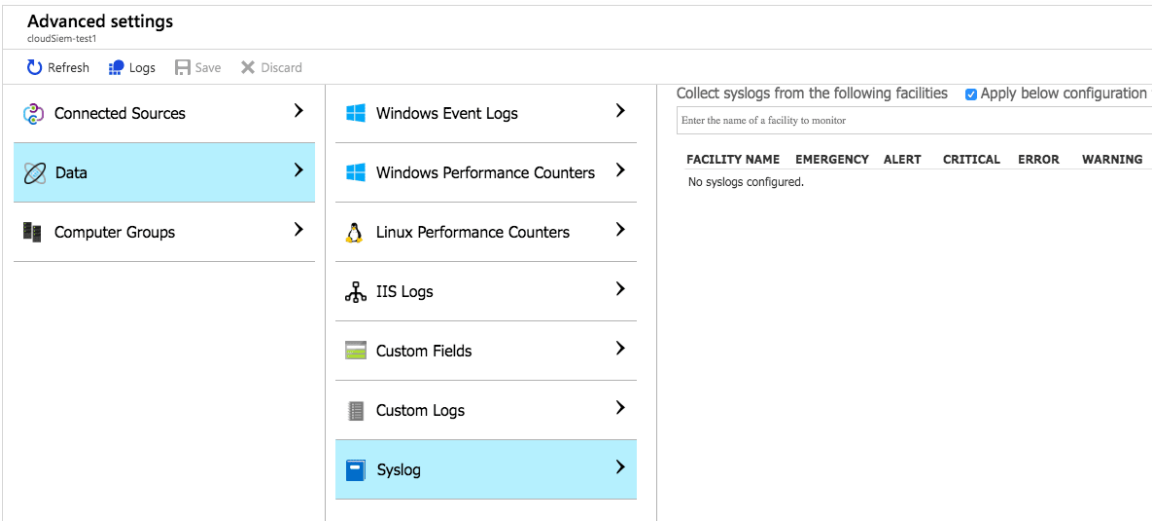


If you want to receive logs from an Azure VM you can select the **Syslog Connector** and pick the VM that you want to use:

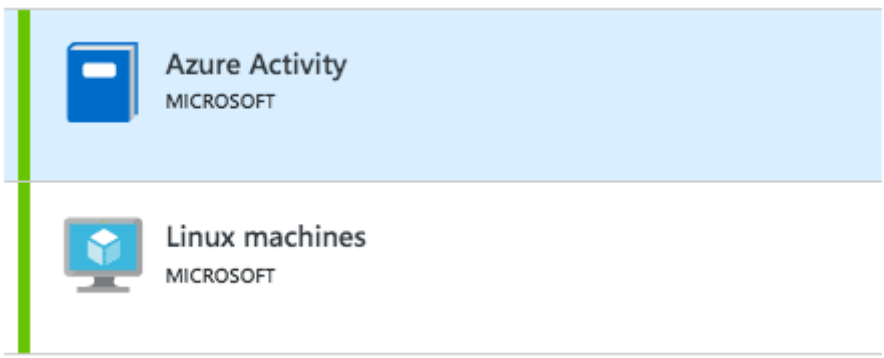
Deploy the Linux agent for example in "Zeek" VM



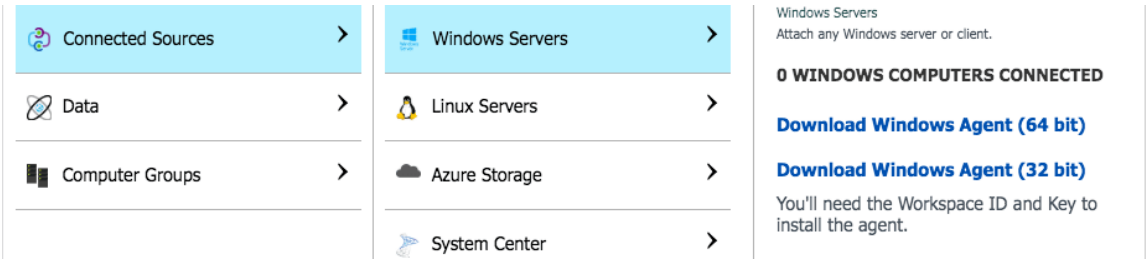
Go to "Advanced Settings" - \> Data - \> Syslog - \> select Apply below configuration to my machines



And now you are connected the Linux Machine

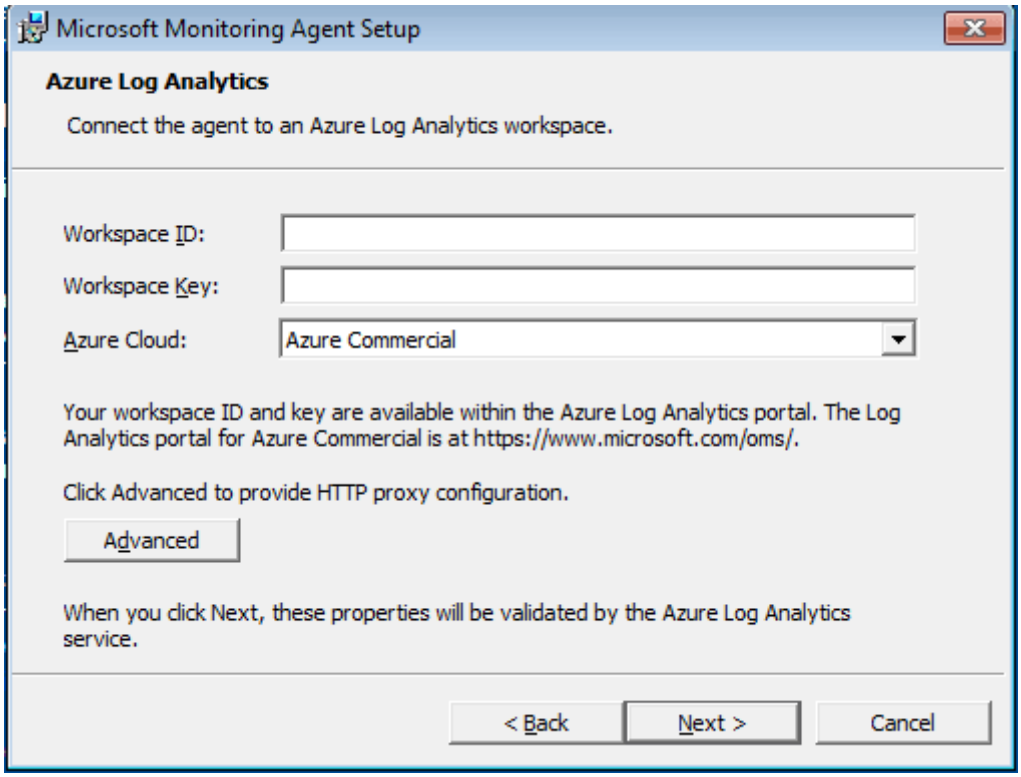


If you want to receive logs from a windows machine: Go to "Advanced Settings" - \> Connected Sources and select "Windows Servers". Then download the Windows agent installation binary

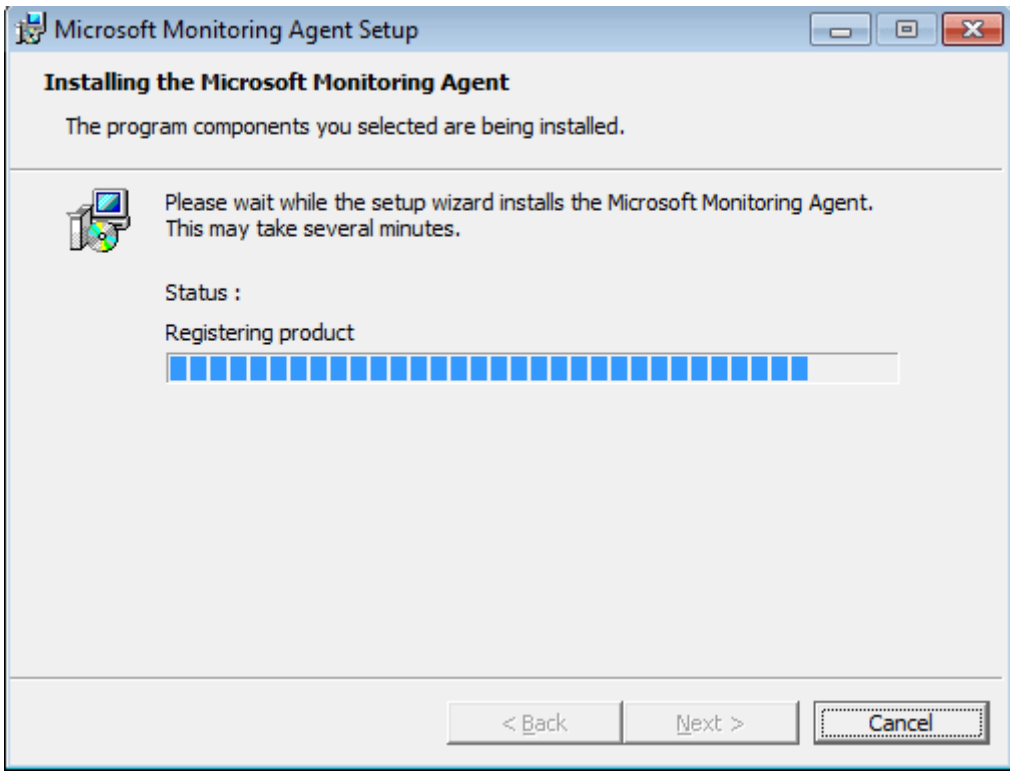


Open your Windows machine (in my case **Windows 7 x32**) and install the agent. Click **Next**

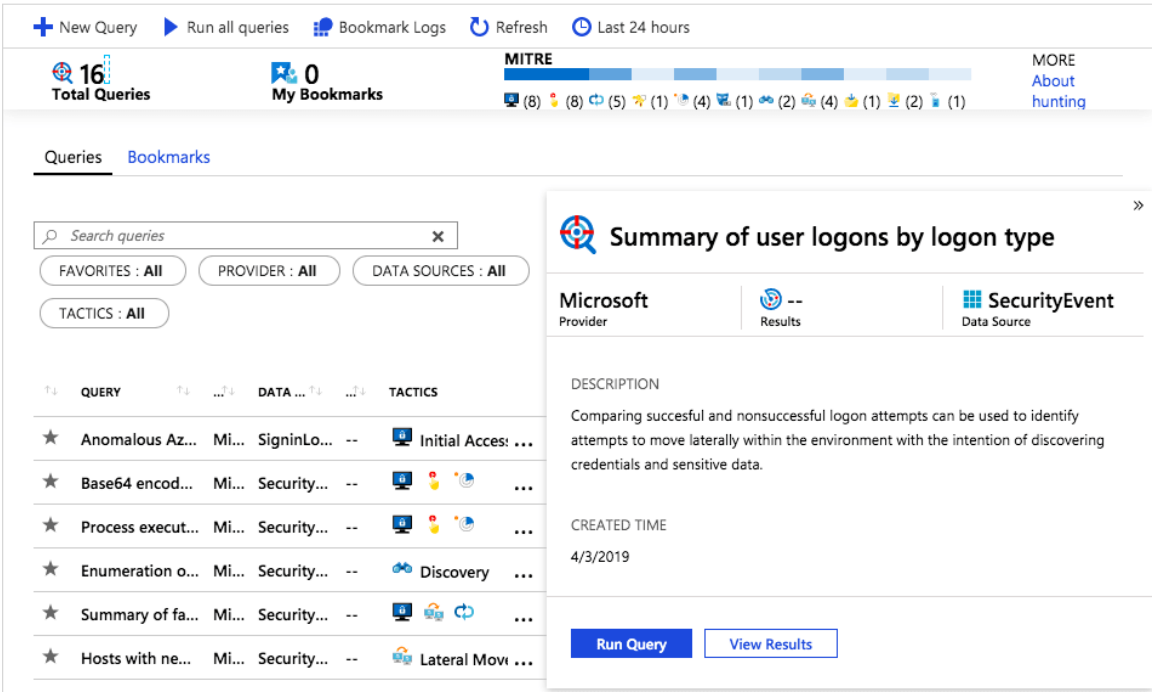
Add your **ID** and **Key** (You will find them in Windows servers dashboard)



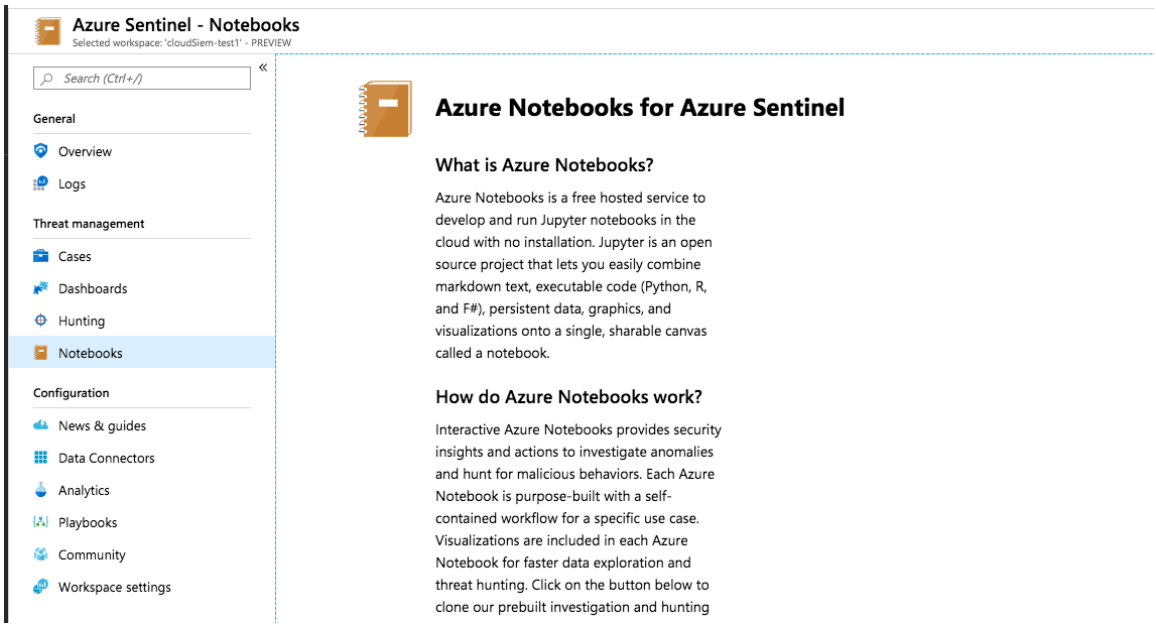
Click **Next** and you are done



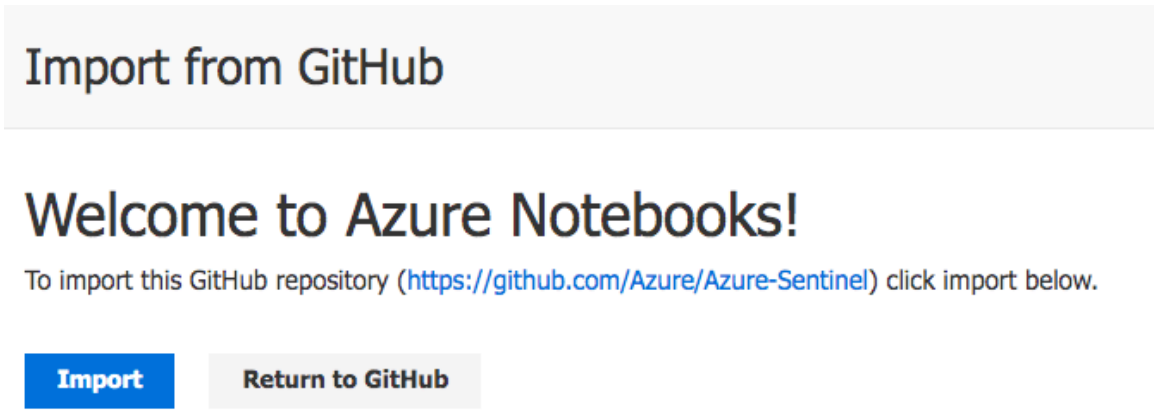
Now it is hunting time! Go to your Sentinel page and select **Hunting** and you will be able to type your own hunting queries using KQL Azure query language.



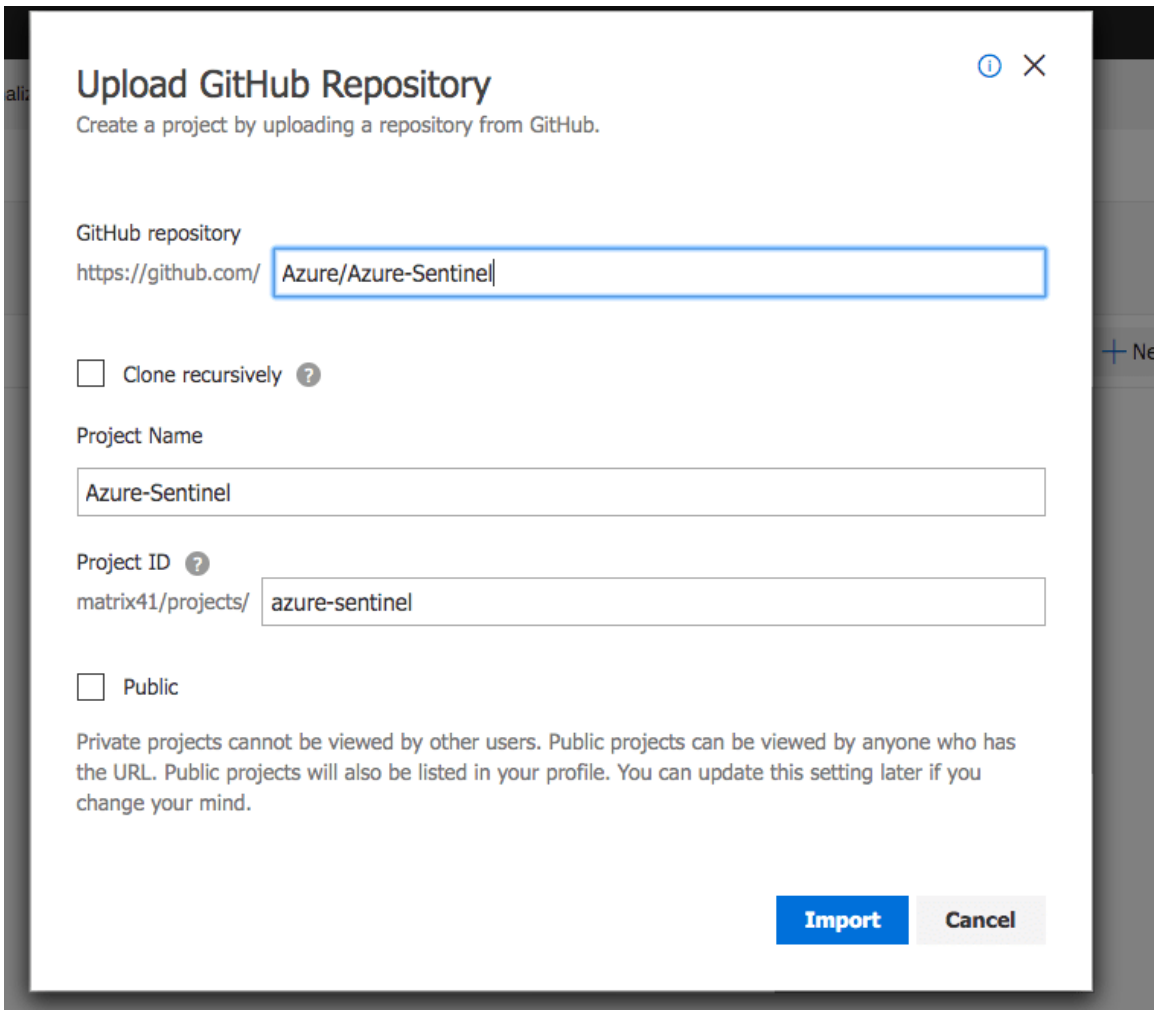
You can also use and create your own Notebooks



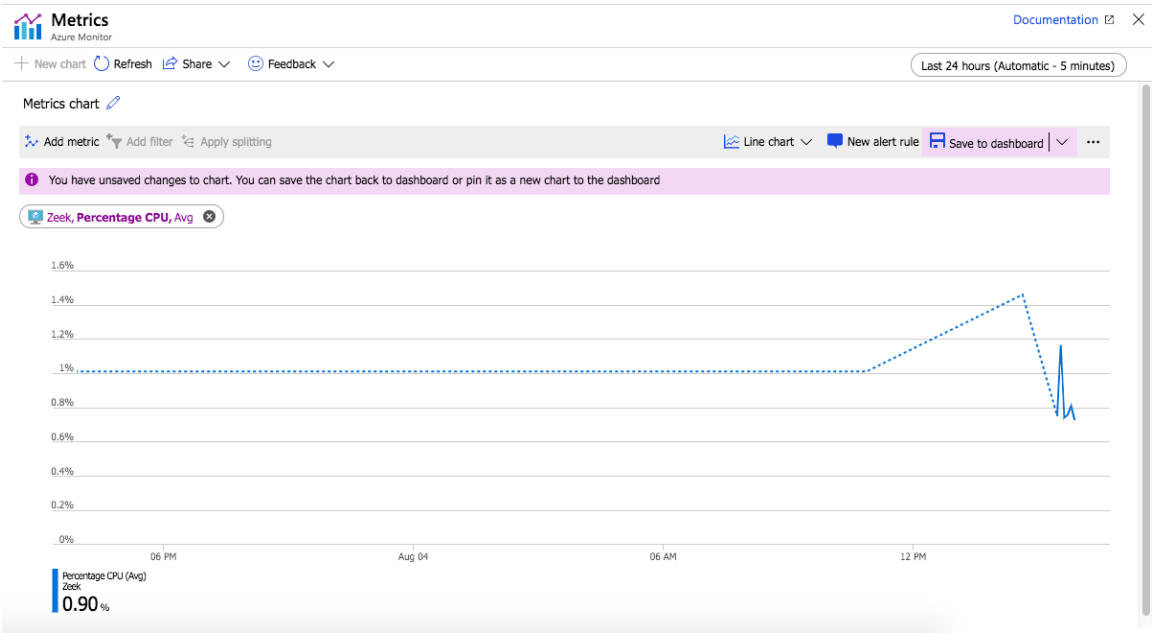
You can use some pre-made hunting notebooks delivered by Azure. **Click Import**



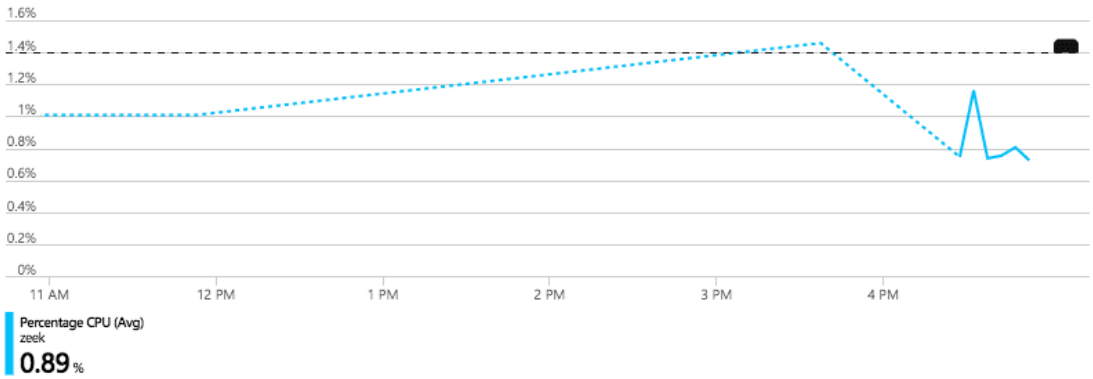
and you will upload them directly from the official Sentinel GitHub account:

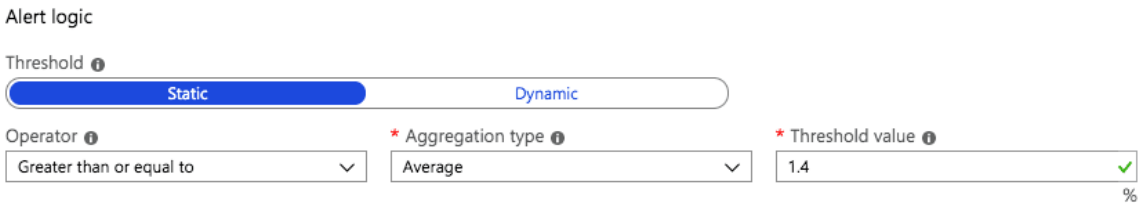


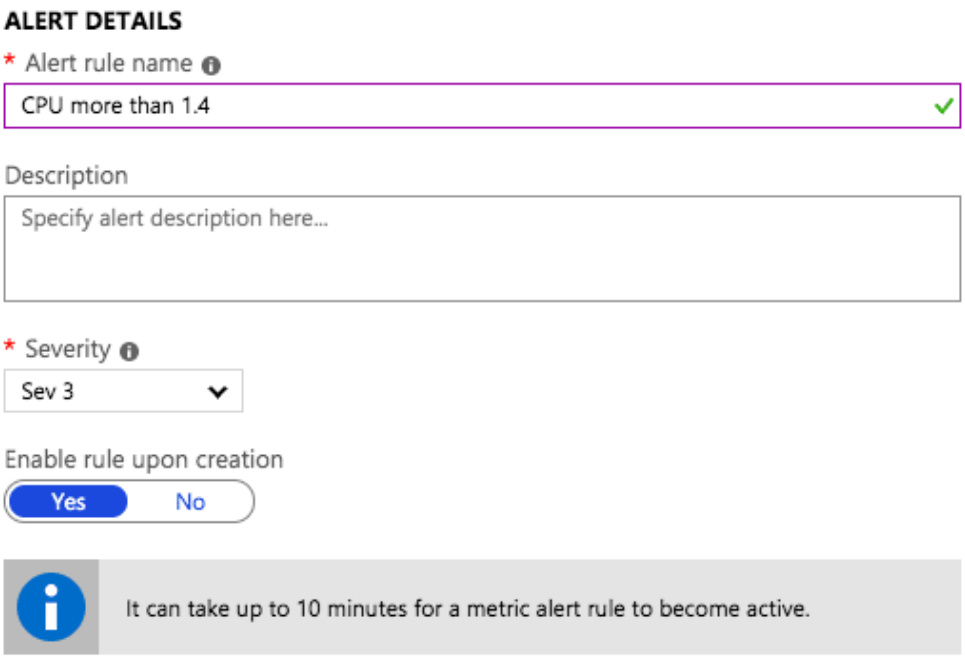
The Sentinel dashboards are highly customizable. In other words, you add any visualisation you want. In this example i added a CPU visualization



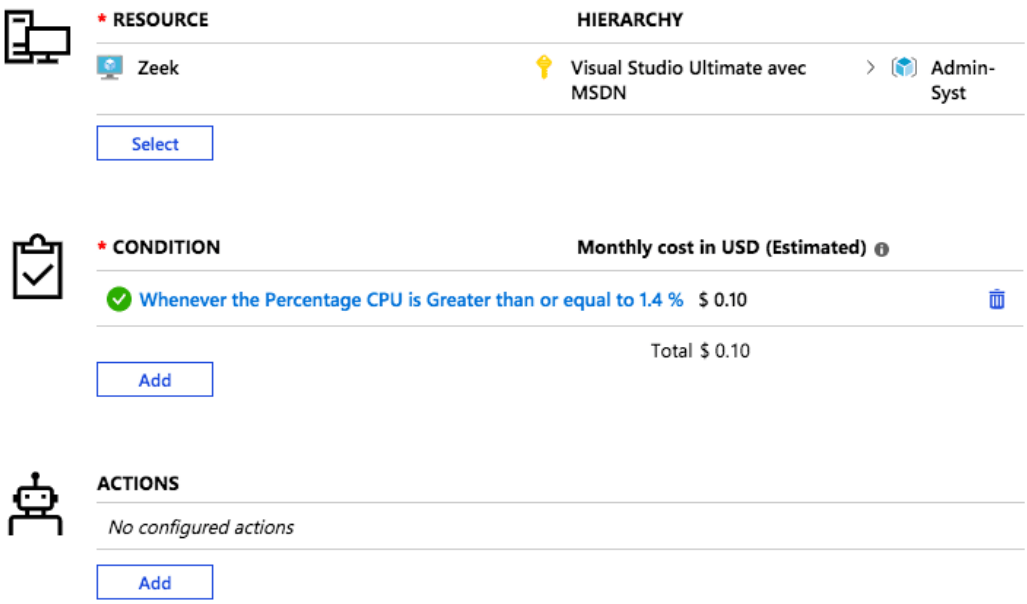
You can even add your alert/detection rules. If you want to do so click "**New alert rule**"



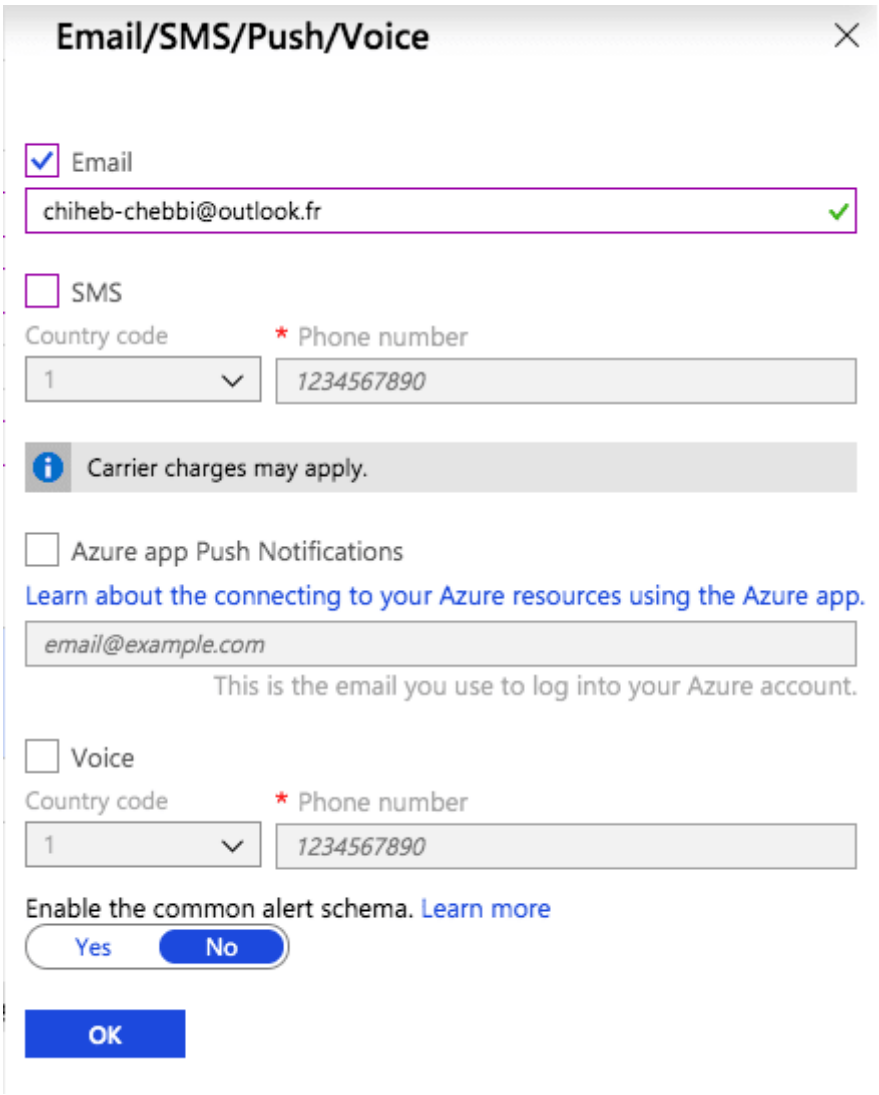




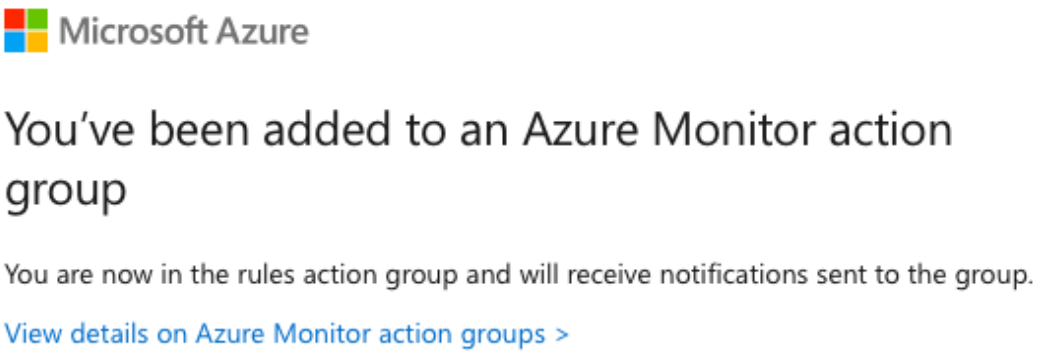
I tried an arbitrary condition for educational purposes **CPU \> 1.4%**



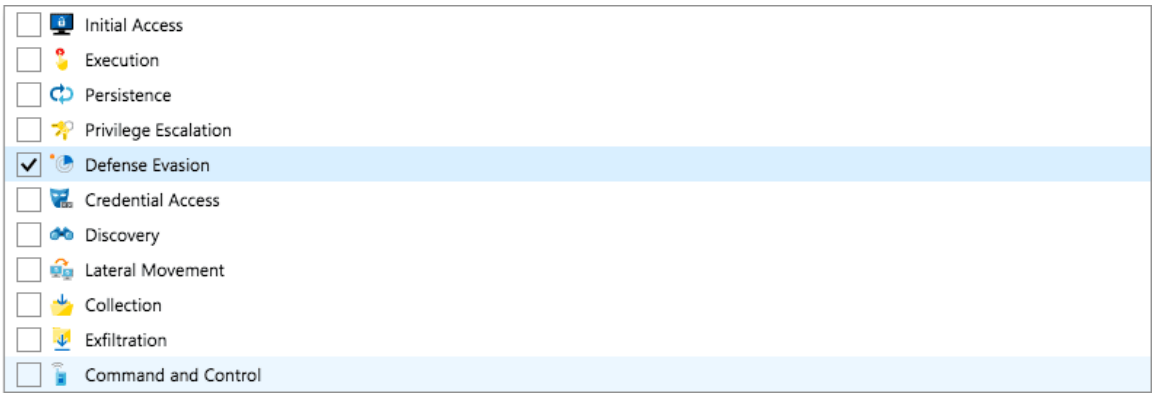
You can also select your action when the condition is performed. In my case, i tried the email notification option



You will receive a confirmation email to check that everything is ok:



When the rule is achieved you will receive an email notification



You can also write your own advanced detection queries with KQL. Go to " **Hunting**" and Click " **New Query**" and create your customized query and also you can identify its connection with MITRE ATT&CK framework.

By now you are ready to start your Hunting mission.

Hands-on Wazuh Host-based Intrusion Detection System (HIDS) Deployment

Hi Peerlysters,

In this article we are going to learn how to deploy a powerful HIDS called "Wazuh"

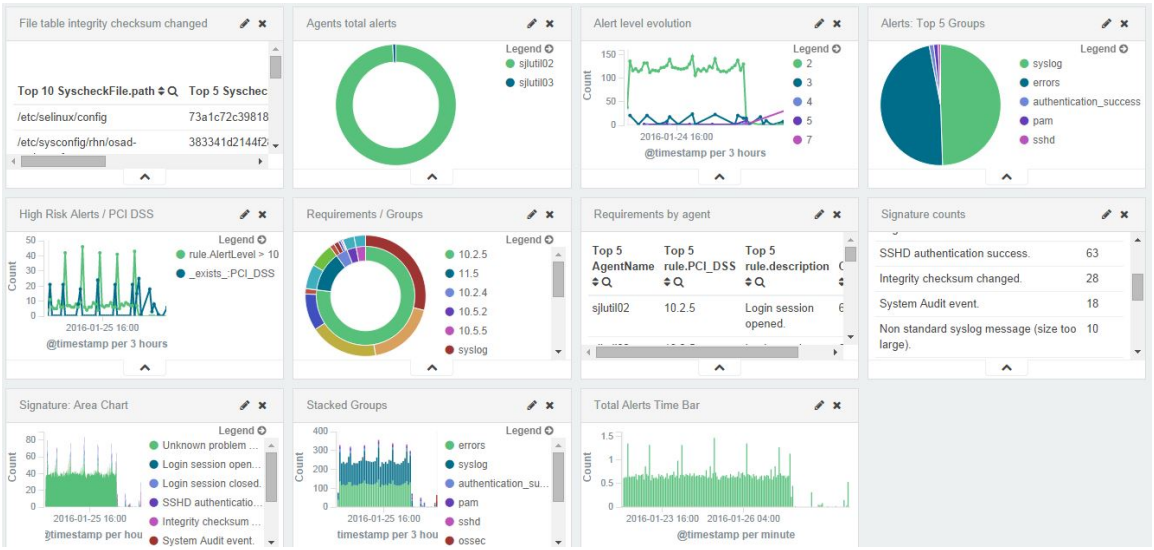


Image Source

What is an intrusion detection system?

Intrusion detection systems are a set of devices or pieces of software that play a huge role in modern organizations to defend against intrusions and malicious activities.We have two major intrusion detection system categories:

- **Host Based Intrusion Detection Systems (HIDS):** they run on the enterprise hosts to detect host attacks
- **Network Based Intrusion Detection Systems (NIDS):** their role is to detect network anomalies by monitoring the inbound and outbound traffic.

The detection can be done using two intrusion detection techniques:

- **Signature based detection technique:** the traffic is compared against a database of signatures of known threats
- Anomaly-based intrusion technique: inspects the traffic based on the behavior of activities.

How to Deploy Wazuh HIDS?

According to its official website: https://wazuh.com

> Wazuh is a free, open source and enterprise-ready security monitoring solution for threat detection, integrity monitoring, incident response and compliance. Wazuh is used to collect, aggregate, index and analyze security data, helping organizations detect intrusions, threats and behavioral anomalies.

Wazuh is used to collect, aggregate, index and analyze security data, helping organizations detect intrusions, threats and behavioral anomalies.

It contains the following components:

- **Wazuh server**
- **Elastic Stack**
- **Wazuh agent**

Now let's explore how to deploy it. For the demonstration i am using a Ubuntu 18.04 VM.

```
sudo apt-get update
```

```
sudo apt-get installcurl apt-transport-https lsb-release gnupg2
```

```
azureuser@Wazuh:~$ sudo apt-get install curl apt-transport-https lsb-release gnu
pg2
Reading package lists... Done
Building dependency tree
Reading state information... Done
lsb-release is already the newest version (9.20170808ubuntu1).
lsb-release set to manually installed.
curl is already the newest version (7.58.0-2ubuntu3.8).
curl set to manually installed.
The following package was automatically installed and is no longer required:
  linux-headers-4.15.0-66
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  apt-transport-https gnupg2
0 upgraded, 2 newly installed, 0 to remove and 30 not upgraded.
Need to get 6360 B of archives.
After this operation, 205 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://azure.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 apt-t
ransport-https all 1.6.12 [1692 B]
Get:2 http://azure.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 gnupg
2 all 2.2.4-1ubuntu1.2 [4668 B]
Fetched 6360 B in 0s (128 kB/s)
Selecting previously unselected package apt-transport-https.
```

Install the GPG key:

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add
-
```

```
azureuser@Wazuh:~$ sudo curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | s
udo  apt-key add -
OK
```

Add the repository

```
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a
/etc/apt/sources.list.d/elastic-7.x.list
```

```
azureuser@Wazuh:~$ sudo echo "deb https://packages.wazuh.com/3.x/apt/ stable mai
n" | sudo tee -a /etc/apt/sources.list.d/wazuh.list
deb https://packages.wazuh.com/3.x/apt/ stable main
```

Update the package information:

```
sudo apt-get update
```

```
azureuser@Wazuh:~$ sudo apt-get update
Hit:1 http://azure.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:3 http://azure.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:4 http://azure.archive.ubuntu.com/ubuntu bionic-backports InRelease
Get:5 https://packages.wazuh.com/3.x/apt stable InRelease [5997 B]
Get:6 https://packages.wazuh.com/3.x/apt stable/main amd64 Packages [15.3 kB]
Fetched 21.3 kB in 1s (42.3 kB/s)
Reading package lists... Done
```

Installing the Wazuh manager

On your terminal, install the Wazuh manager:

```
sudo apt-get install wazuh-manager
```

```
azureuser@Wazuh:~$ sudo apt-get install wazuh-manager
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  linux-headers-4.15.0-66
Use 'sudo apt autoremove' to remove it.
Suggested packages:
  expect
The following NEW packages will be installed:
  wazuh-manager
0 upgraded, 1 newly installed, 0 to remove and 30 not upgraded.
Need to get 70.7 MB of archives.
After this operation, 339 MB of additional disk space will be used.
Get:1 https://packages.wazuh.com/3.x/apt stable/main amd64 wazuh-manager amd64 3
.10.2-1 [70.7 MB]
Fetched 70.7 MB in 4s (17.8 MB/s)
Selecting previously unselected package wazuh-manager.
(Reading database ... 74430 files and directories currently installed.)
Preparing to unpack .../wazuh-manager_3.10.2-1_amd64.deb ...
Unpacking wazuh-manager (3.10.2-1) ...
Setting up wazuh-manager (3.10.2-1) ...
Processing triggers for systemd (237-3ubuntu10.31) ...
Processing triggers for ureadahead (0.100.0-21) ...
```

Once the process is completed, you can check the service status with:

```
service wazuh-manager status
```

```
● wazuh-manager.service - Wazuh manager
   Loaded: loaded (/etc/systemd/system/wazuh-manager.service; enabled; vendor p
   Active: active (running) since Mon 2019-11-11 09:13:37 UTC; 7min ago
    Tasks: 85 (limit: 9512)
   CGroup: /system.slice/wazuh-manager.service
           ├─29527 /var/ossec/bin/ossec-authd
           ├─29537 /var/ossec/bin/wazuh-db
           ├─29554 /var/ossec/bin/ossec-execd
           ├─29562 /var/ossec/bin/ossec-analysisd
           ├─29571 /var/ossec/bin/ossec-syscheckd
           ├─29581 /var/ossec/bin/ossec-remoted
           ├─29583 /var/ossec/bin/ossec-logcollector
           ├─29606 /var/ossec/bin/ossec-monitord
           └─29612 /var/ossec/bin/wazuh-modulesd

Nov 11 09:13:35 Wazuh env[29444]: Started wazuh-db...
Nov 11 09:13:35 Wazuh env[29444]: Started ossec-execd...
Nov 11 09:13:35 Wazuh env[29444]: Started ossec-analysisd...
Nov 11 09:13:35 Wazuh env[29444]: Started ossec-syscheckd...
Nov 11 09:13:35 Wazuh env[29444]: Started ossec-remoted...
Nov 11 09:13:35 Wazuh env[29444]: Started ossec-logcollector...
Nov 11 09:13:35 Wazuh env[29444]: Started ossec-monitord...
Nov 11 09:13:35 Wazuh env[29444]: Started wazuh-modulesd...
lines 1-23
```

Installing the Wazuh API:

NodeJS \>= 4.6.1 is required in order to run the Wazuh API.

```
sudo curl -sL https://deb.nodesource.com/setup_8.x | sudo  bash -
```

```
azureuser@Wazuh:~$ sudo curl -sL https://deb.nodesource.com/setup_8.x |sudo  bas
h -

## Installing the NodeSource Node.js 8.x LTS Carbon repo...


## Populating apt-get cache...

+ apt-get update
Hit:1 http://azure.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Hit:3 http://azure.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:4 https://packages.wazuh.com/3.x/apt stable InRelease
Hit:5 http://azure.archive.ubuntu.com/ubuntu bionic-backports InRelease
Fetched 88.7 kB in 0s (187 kB/s)
Reading package lists... Done

## Confirming "bionic" is supported...

+ curl -sLf -o /dev/null 'https://deb.nodesource.com/node_8.x/dists/bionic/Relea
se'

## Adding the NodeSource signing key to your keyring...
```

and then, install NodeJS:

```
sudo apt-get install nodejs
```

```
azureuser@Wazuh:~$ sudo apt-get install nodejs
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  linux-headers-4.15.0-66
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  nodejs
0 upgraded, 1 newly installed, 0 to remove and 30 not upgraded.
Need to get 13.6 MB of archives.
After this operation, 64.5 MB of additional disk space will be used.
Get:1 https://deb.nodesource.com/node_8.x bionic/main amd64 nodejs amd64 8.16.2-
1nodesource1 [13.6 MB]
Fetched 13.6 MB in 0s (42.6 MB/s)
Selecting previously unselected package nodejs.
(Reading database ... 86918 files and directories currently installed.)
Preparing to unpack .../nodejs_8.16.2-1nodesource1_amd64.deb ...
Unpacking nodejs (8.16.2-1nodesource1) ...
Setting up nodejs (8.16.2-1nodesource1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
```

Install the Wazuh API:

```
sudo apt-get install wazuh-api
```

```
azureuser@Wazuh:~$ sudo apt-get install wazuh-api
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  linux-headers-4.15.0-66
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  wazuh-api
0 upgraded, 1 newly installed, 0 to remove and 30 not upgraded.
Need to get 2223 kB of archives.
After this operation, 9929 kB of additional disk space will be used.
Get:1 https://packages.wazuh.com/3.x/apt stable/main amd64 wazuh-api amd64 3.10.
2-1 [2223 kB]
Fetched 2223 kB in 2s (1335 kB/s)
Selecting previously unselected package wazuh-api.
(Reading database ... 90403 files and directories currently installed.)
Preparing to unpack .../wazuh-api_3.10.2-1_amd64.deb ...
Unpacking wazuh-api (3.10.2-1) ...
Setting up wazuh-api (3.10.2-1) ...
Synchronizing state of wazuh-api.service with SysV service script with /lib/syst
emd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable wazuh-api
Processing triggers for systemd (237-3ubuntu10.31) ...
```

Once the process is complete, you can check the service status with:

`sudo service wazuh-api status`

```
azureuser@Wazuh:~$ sudo service wazuh-api status
● wazuh-api.service - Wazuh API daemon
   Loaded: loaded (/etc/systemd/system/wazuh-api.service; enabled; vendor preset
   Active: active (running) since Mon 2019-11-11 09:25:04 UTC; 54s ago
     Docs: https://documentation.wazuh.com/current/user-manual/api/index.html
 Main PID: 37540 (nodejs)
    Tasks: 10 (limit: 9512)
   CGroup: /system.slice/wazuh-api.service
           └─37540 /usr/bin/nodejs /var/ossec/api/app.js

Nov 11 09:25:04 Wazuh systemd[1]: Started Wazuh API daemon.
lines 1-10/10 (END)
```

Installing Filebeat

`apt-get install filebeat=7.4.2`

```
azureuser@Wazuh:~$ sudo apt-get install filebeat=7.4.2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  linux-headers-4.15.0-66
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  filebeat
0 upgraded, 1 newly installed, 0 to remove and 30 not upgraded.
Need to get 24.2 MB of archives.
After this operation, 78.1 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/7.x/apt stable/main amd64 filebeat a
md64 7.4.2 [24.2 MB]
Fetched 24.2 MB in 1s (46.6 MB/s)
Selecting previously unselected package filebeat.
(Reading database ... 92997 files and directories currently installed.)
Preparing to unpack .../filebeat_7.4.2_amd64.deb ...
Unpacking filebeat (7.4.2) ...
Setting up filebeat (7.4.2) ...
Processing triggers for systemd (237-3ubuntu10.31) ...
Processing triggers for ureadahead (0.100.0-21) ...
```

This is pre-configuration to forward Wazuh alerts to Elasticsearch

```
curl -so /etc/filebeat/filebeat.yml
https://raw.githubusercontent.com/wazuh/wazuh/v3.11.4/extensions/filebeat/7.x/file
beat.yml
```

Download the alerts template for Elasticsearch

```
curl -so /etc/filebeat/wazuh-template.json
https://raw.githubusercontent.com/wazuh/wazuh/v3.11.4/extensions/elasticsearch/7.x
/wazuh-template.json
```

Download the Wazuh module for Filebeat:

```
curl -s https://packages.wazuh.com/3.x/filebeat/wazuh-filebeat-0.1.tar.gz | sudo
tar -xvz -C /usr/share/filebeat/module
```

```
azureuser@Wazuh:~$ sudo curl -s https://packages.wazuh.com/3.x/filebeat/wazuh-fi
lebeat-0.1.tar.gz | sudo tar -xvz -C /usr/share/filebeat/module
wazuh/alerts/
wazuh/alerts/manifest.yml
wazuh/alerts/config/
wazuh/alerts/config/alerts.yml
wazuh/alerts/ingest/
wazuh/alerts/ingest/pipeline.json
wazuh/archives/
wazuh/archives/manifest.yml
wazuh/archives/config/
wazuh/archives/config/archives.yml
wazuh/archives/ingest/
wazuh/archives/ingest/pipeline.json
wazuh/module.yml
azureuser@Wazuh:~$
azureuser@Wazuh:~$
azureuser@Wazuh:~$ sudo vi /etc/filebeat/filebeat.yml
azureuser@Wazuh:~$ sudo update-rc.d filebeat defaults 95 10
azureuser@Wazuh:~$ sudo service filebeat start
azureuser@Wazuh:~$ sudo apt-get install elasticsearch=7.4.2
```

`sudo vi /etc/filebeat/filebeat.yml`

```
# Wazuh - Filebeat configuration file
filebeat.modules:
  - module: wazuh
    alerts:
      enabled: true
    archives:
      enabled: false

setup.template.json.enabled: true
setup.template.json.path: '/etc/filebeat/wazuh-template.json'
setup.template.json.name: 'wazuh'
setup.template.overwrite: true
setup.ilm.enabled: false

output.elasticsearch.hosts: ['localhost:9200']
~
```

Enable and start the Filebeat service:

`sudo update-rc.d filebeat defaults 95 10`

`sudo service filebeat start`

Installing Elastic Stack

Elasticsearch is a powerful open source distributed, RESTful, JSON-based search engine.You can see it as a search server.It is a NoSQL database.To install elasticsearch we need to make sure that we are already installed Java.

`sudo apt-get install elasticsearch=7.4.2`

`sudo vi /etc/elasticsearch/elasticsearch.yml`

```
node.name: node-1
network.host: ["0.0.0.0"]
http.port: 9200
discovery.seed_hosts: []
cluster.initial_master_nodes: ["node-1"]
```

`sudo update-rc.d elasticsearch defaults 95 10`

`sudo service elasticsearch start`

```
azureuser@Wazuh:~$ sudo vi /etc/elasticsearch/elasticsearch.yml
azureuser@Wazuh:~$ sudo update-rc.d elasticsearch defaults 95 10
azureuser@Wazuh:~$ sudo service elasticsearch start
azureuser@Wazuh:~$ sudo service elasticsearch status
● elasticsearch.service - Elasticsearch
   Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; disabled; vend
   Active: active (running) since Mon 2019-11-11 09:50:41 UTC; 11s ago
     Docs: http://www.elastic.co
 Main PID: 45278 (java)
    Tasks: 40 (limit: 9512)
   CGroup: /system.slice/elasticsearch.service
           ├─45278 /usr/share/elasticsearch/jdk/bin/java -Xms1g -Xmx1g -XX:+UseC
           └─45396 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-x86

Nov 11 09:50:23 Wazuh systemd[1]: Starting Elasticsearch...
Nov 11 09:50:24 Wazuh elasticsearch[45278]: OpenJDK 64-Bit Server VM warning: Op
Nov 11 09:50:41 Wazuh systemd[1]: Started Elasticsearch.
lines 1-13/13 (END)
```

Once Elasticsearch is up and running, it is recommended to load the Filebeat template. Run the following command where Filebeat was installed:

```
sudo filebeat setup --index-management -E setup.template.json.enabled=false
```

```
azureuser@Wazuh:~$ sudo filebeat setup --index-management -E setup.template.json
.enabled=false
ILM policy and write alias loading not enabled.
Index setup finished.
azureuser@Wazuh:~$
```

Installing Kibana

Kibana is a Web interface for searching and visualizing logs. It is a data-log dashboard. It contains pie charts, bars, heat maps, bubble charts and scatter plots. It is an amazing solution to visualize your data and detect any unusual patterns

```
apt-get install kibana=7.4.2
```

```
azureuser@Wazuh:~$ sudo apt-get install kibana=7.4.2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  linux-headers-4.15.0-66
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  kibana
0 upgraded, 1 newly installed, 0 to remove and 30 not upgraded.
Need to get 259 MB of archives.
After this operation, 722 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/7.x/apt stable/main amd64 kibana amd
64 7.4.2 [259 MB]
Fetched 259 MB in 5s (47.5 MB/s)
Selecting previously unselected package kibana.
(Reading database ... 110394 files and directories currently installed.)
Preparing to unpack .../kibana_7.4.2_amd64.deb ...
Unpacking kibana (7.4.2) ...
Setting up kibana (7.4.2) ...
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for systemd (237-3ubuntu10.31) ...
```

Install the Wazuh app plugin for Kibana

```
sudo -u kibana bin/kibana-plugin install
https://packages.wazuh.com/wazuhapp/wazuhapp-3.11.4_7.6.1.zip
```

```
sudo vi /etc/kibana/kibana.yml
```

```
server.port: 5601
server.host: 0.0.0.0
elasticsearch.hosts: ["http://localhost:9200"]
```

```
sudo update-rc.d kibana defaults 95 10
```

```
service kibana start
```

Transform data with Logstash (Optional)

Logstash is an open source to collect,parse and transform logs.

```
sudo apt-get install logstash=1:7.4.2-1
```

`sudo systemctl daemon-reload`

`sudo systemctl enable logstash`

```
azureuser@Wazuh:~$ sudo apt-get install logstash=1:7.4.2-1
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  linux-headers-4.15.0-66
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  logstash
0 upgraded, 1 newly installed, 0 to remove and 30 not upgraded.
Need to get 175 MB of archives.
After this operation, 304 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/7.x/apt stable/main amd64 logstash a
ll 1:7.4.2-1 [175 MB]
Fetched 175 MB in 5s (35.4 MB/s)
Selecting previously unselected package logstash.
(Reading database ... 219439 files and directories currently installed.)
Preparing to unpack .../logstash_1%3a7.4.2-1_all.deb ...
Unpacking logstash (1:7.4.2-1) ...
Setting up logstash (1:7.4.2-1) ...
Using provided startup.options file: /etc/logstash/startup.options
/usr/share/logstash/vendor/bundle/jruby/2.5.0/gems/pleaserun-0.0.30/lib/pleaseru
n/platform/base.rb:112: warning: constant ::Fixnum is deprecated
Successfully created system startup script for Logstash
```

Download the Wazuh configuration file for Logstash

sudo systemctl restart logstash

sudo vi /etc/filebeat/filebeat.yml\</a

Configure the Filebeat instance, change the events destination from Elasticsearch instance to the Logstash instance.

Disable Elasticsearch Output:

Add:

output.logstash.hosts: ["localhost:5000"]

sudo systemctl restart filebeat

Check if Logstash is reachable from Filebeat.

sudo filebeat test output

```
[azureuser@Wazuh:~$ sudo curl -so /etc/logstash/conf.d/01-wazuh.conf https://raw.
githubusercontent.com/wazuh/wazuh/v3.10.2/extensions/logstash/7.x/01-wazuh-remot
e.conf
[azureuser@Wazuh:~$ sudo systemctl restart logstash
[azureuser@Wazuh:~$ sudo vi /etc/filebeat/filebeat.yml
[azureuser@Wazuh:~$ sudo systemctl restart filebeat
[azureuser@Wazuh:~$ sudo filebeat test output
logstash: localhost:5000...
  connection...
    parse host... OK
    dns lookup... OK
    addresses: 127.0.0.1
    dial up... OK
  TLS... WARN secure connection disabled
  talk to server... OK
```

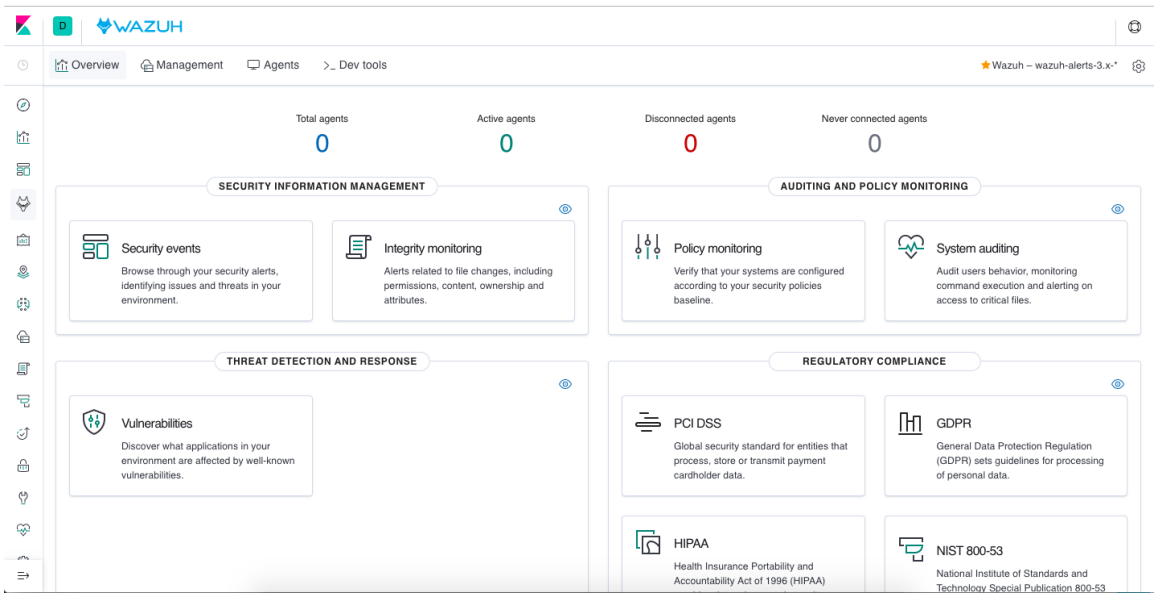
Replace the default credentials with your desired username where myUsername is shown below to protect your Wazuh API

```
[azureuser@Wazuh:~$ sudo bash
[root@Wazuh:~# cd  /var/ossec/api/configuration/auth
[root@Wazuh:/var/ossec/api/configuration/auth# ls
htpasswd  user
[root@Wazuh:/var/ossec/api/configuration/auth# node htpasswd -c user myUserName
[New password:
[Re-type new password:
Adding password for user myUserName.
[root@Wazuh:/var/ossec/api/configuration/auth# service wazuh-api restart
root@Wazuh:/var/ossec/api/configuration/auth#
```

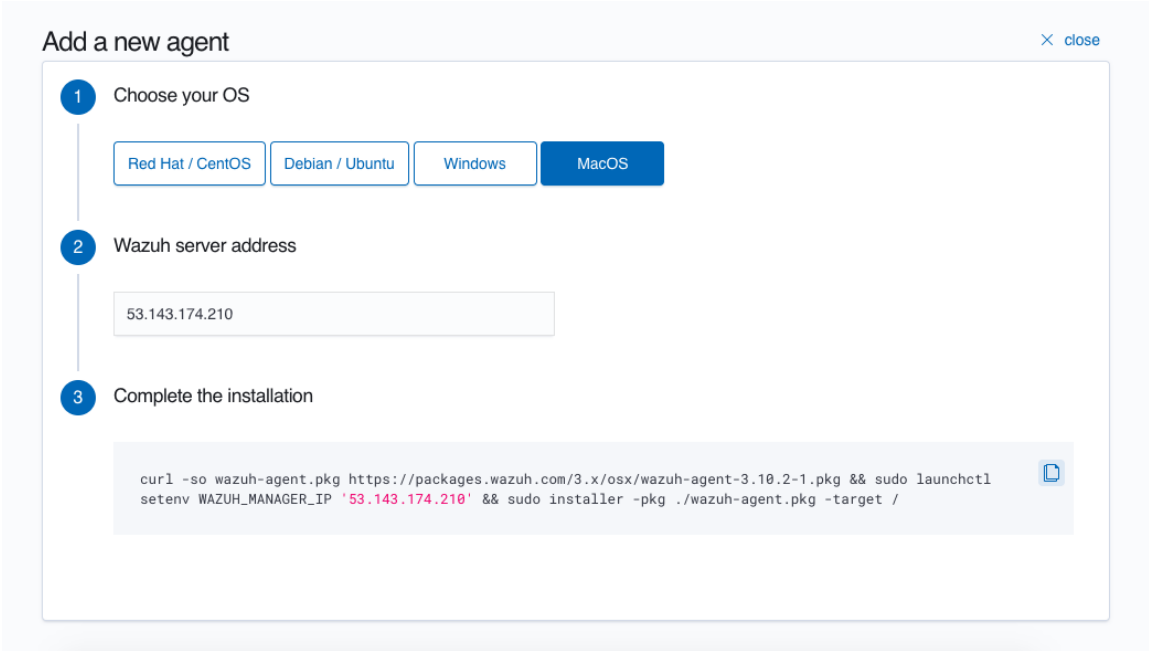
More information: https://documentation.wazuh.com/3.3/installation-guide/installing-elastic-stack/connect_wazuh_app.html

Open a web browser and go to the Elastic Stack server's IP address on port 5601 (default Kibana port). Then, from the left menu, go to the Wazuh App.

Click on "Add new API" and fill the API fields. If everything goes fine, you will get this main Wazuh dashboard.



To add new agent just select the OS, curl the package and install it:



Threat Intelligence Fundamentals

What is a threat?

By definition, a threat is a potential danger for the enterprise assets that could harm these systems. In many cases, there is confusion between the three terms Threat, Vulnerability and Risk; the first term, as I explained before, is a potential danger while a Vulnerability is a known weakness or a gap in an asset. A risk is a result of a threat exploiting a vulnerability. In other words, you can see it as an intersection between the two previous terms. The method used to attack an asset is called a Threat Vector.

There are three main types of threats: * Natural threats * Unintentional threats * Intentional threats

What is an advanced Persistent Threat (APT)?

Wikipedia defines an "Advanced Persistent Threat" as follows:

> "An advanced persistent threat is a stealthy computer network threat actor, typically a nation-state or state-sponsored group, which gains unauthorized access to a computer network and remains undetected for an extended period"

To explore some APTs Check this great resource by: FireEye

What is Threat Intelligence?

> “Cyber threat intelligence is information about threats and threat actors that helps mitigate harmful events in cyberspace. Cyber threat intelligence sources include open source intelligence, social media intelligence, human Intelligence, technical intelligence or intelligence from the deep and dark web "[Source: Wikipedia]

In other words, intelligence differs from data and information as completing the full picture.

Threat Intelligence goes through the following steps:

1. Planning and direction
2. Collection
3. Processing and exploitation
4. Analysis and production
5. Dissemination and integration

What are the Indicators of compromise (IOCs)?

Indicators of compromise are pieces of information about a threat that can be used to detect intrusions such as MD5 hashed, URLs, IP addresses and so on.

These pieces can be shared accross different organizations thanks to bodies like: * Information Sharing and Analysis Centers (ISACs) * Computers emergency response teams (CERTs) * Malware Information Sharing Platform (MISP)

To facilitate the sharing/collecting/analyzing processes these IOCs usually respect and follow certain formats and protocols such as:

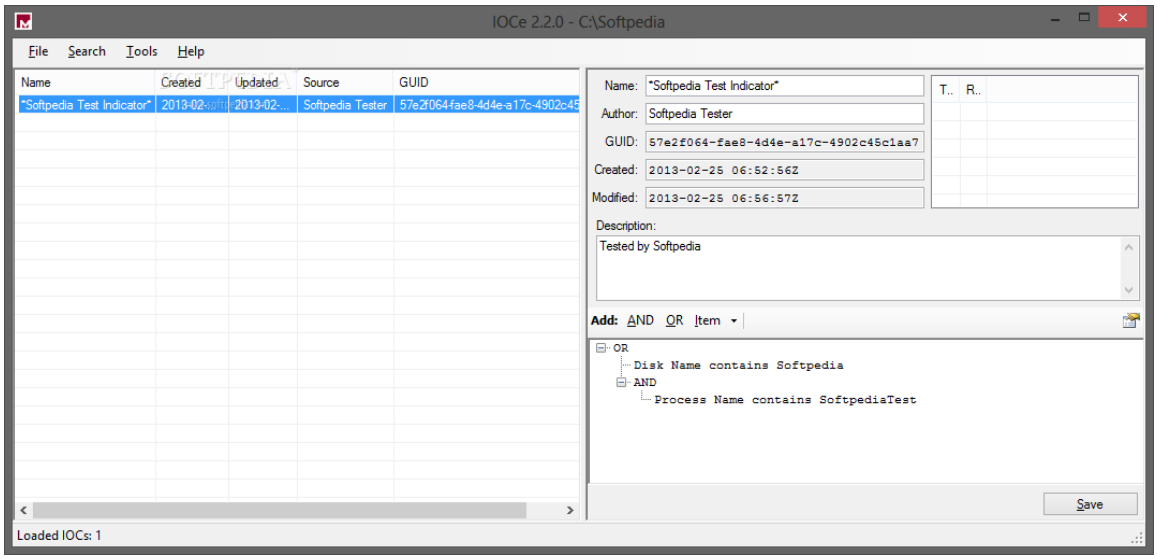
- OpenIOC
- Structured Threat Information eXpression (STIX)
- Trusted Automated Exchange of Intelligence Information (TAXII)

For example, this is the IOC STIX representation of Wannacry ransomware:

```
▼<stix:Indicators>
  ▼<stix:Indicator id="indicator-2cc6ee0f-3c34-11e7-846c-64006a8636ca" timestamp="2017-05-
   19T01:50:28.526000+00:00" xsi:type="indicator:IndicatorType">
     <indicator:Title>Malicious File Indicator</indicator:Title>
     <indicator:Type xsi:type="stixVocabs:IndicatorTypeVocab-1.1">File Hash
     Watchlist</indicator:Type>
    ▼<indicator:Description>
       Based on US-CERT analysis, this hash may be associated with WannaCry Ransomware
       activity.
     </indicator:Description>
    ▼<indicator:Observable id="NCCIC:Observable-7f142976-87d0-4cba-a0c8-0ad36be8dc1f">
      ▼<cybox:Object id="NCCIC:Object-2cc6ee10-3c34-11e7-99b5-64006a8636ca">
        ▼<cybox:Properties xsi:type="FileObj:FileObjectType">
           <FileObj:File_Name condition="Equals">qeriuwjhrf</FileObj:File_Name>
           <FileObj:Size_In_Bytes condition="Equals">3514368</FileObj:Size_In_Bytes>
          ▼<FileObj:Hashes>
            ▼<cyboxCommon:Hash>
               <cyboxCommon:Type condition="Equals" xsi:type="cyboxVocabs:HashNameVocab-
               1.0">MD5</cyboxCommon:Type>
               <cyboxCommon:Simple_Hash_Value
               condition="Equals">3175E4BA26E1E75E52935009A526002C</cyboxCommon:Simple_Hash_\
             </cyboxCommon:Hash>
            ▼<cyboxCommon:Hash>
               <cyboxCommon:Type condition="Equals" xsi:type="cyboxVocabs:HashNameVocab-
               1.0">SHA1</cyboxCommon:Type>
               <cyboxCommon:Simple_Hash_Value
               condition="Equals">5D68E2779E2CCCEE49188363BE6CDDBB0BAC7053</cyboxCommon:Simpl
             </cyboxCommon:Hash>
```

To help you create and edit your indicators of compromise you can use, for rxample, IOC editor by Fireeye. You can find it here: This is its user guide:

You can simply create your Indicators of compromise using a graphical interface:



It gives you also the ability to compare IOCs

How to Install and use The Hive Project in Incident Management

In this module, we are going to explore a great incident management platform called "TheHive Project."

Figure

The Hive Project

According to its official Github repository:

Figure

> "TheHive is a scalable 4-in-1 open source and free security incident response platform designed to make life easier for SOCs, CSIRTs, CERTs and any information security practitioner dealing with security incidents that need to be investigated and acted upon swiftly. Thanks to Cortex, our powerful free and open-source analysis engine, you can analyze (and triage) observables at scale using more than 100 analyzers."

To deploy the project you need these hardware requirements:

- *8vCPU*
- *8 GB of RAM*
- *60 GB of disk*

Now let's explore how to install the project:

First, you need to install Java:

```
sudo   apt-get install openjdk-11-jre-headless
```

Add the sources:

```
echo 'deb https://dl.bintray.com/thehive-project/debian-stable any main' | sudo
tee -a /etc/apt/sources.list.d/thehive-project.list
```

```
curl https://raw.githubusercontent.com/TheHive-Project/TheHive/master /PGP-PUBLIC-
KEY | sudo apt-key add -
```

Update the system:

```
sudo apt-get update
```

Install Elasticsearch

Figure

```
apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-key D88E42B4
```

```
echo "deb https://artifacts.elastic.co/packages/5.x/apt stable main" | tee -a/etc/apt/sources.list.d/elastic-5.x.list
```

```
apt install apt-transport-https
```

```
apt update
```

```
sudo apt install elasticsearch
```

Install "The Hive"

```
sudo apt-get install thehive
```

```
sudo mkdir /etc/thehive
```

```
sudo mkdir /etc/thehive
(cat << _EOF_
# Secret key
# ~~~~~
# The secret key is used to secure cryptographics functions.
# If you deploy your application to several instances be sure to
use the same key!
play.http.secret.key="<ADD A RANDOM STRING HERE>"
_EOF_
) | sudo tee -a /etc/thehive/application.conf
```

```
sudo systemctl enable thehive
```

```
sudo service thehive start
```

Now go to your browser and type:

http://YOUR_SERVER_ADDRESS:9000/

If you want to try it before installing it on your server you download the training VM. You can find it here:

https://drive.google.com/file/d/1KXL7kzH7Pc2jSL2o1m1_RwVc3FGw-ixQ/view

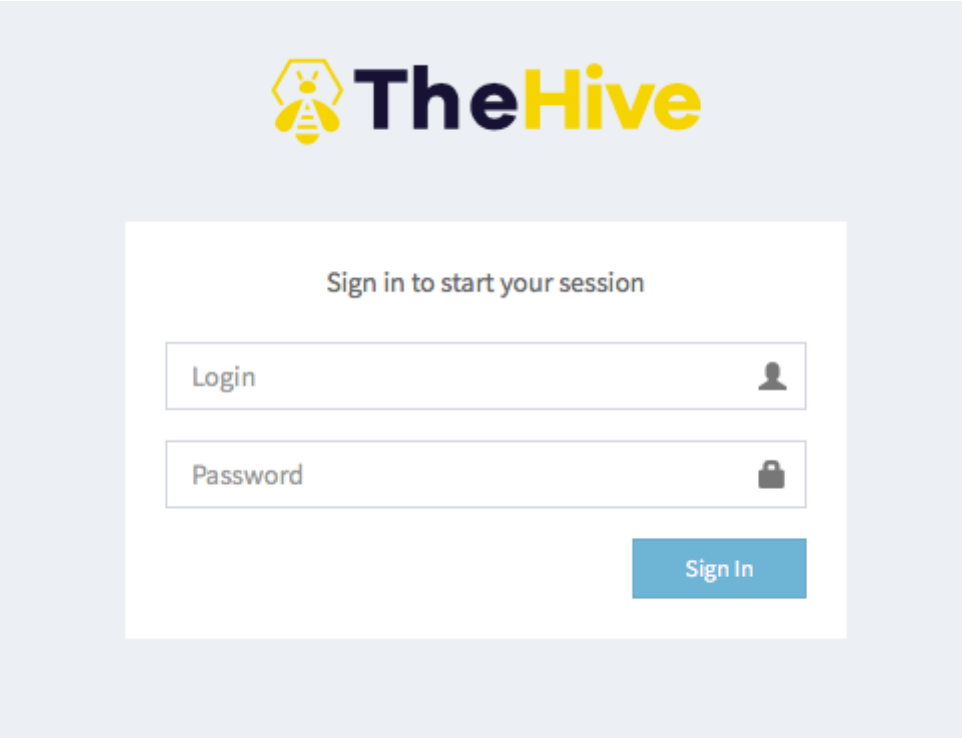
Once you download it, open it with your virtual machine

```
Ubuntu 18.04.1 LTS thehive-training tty1

---

IP address: 192.168.43.188

---

TheHive -> http://192.168.43.188:9000
Cortex  -> http://192.168.43.188:9001

----
thehive-training login:
```

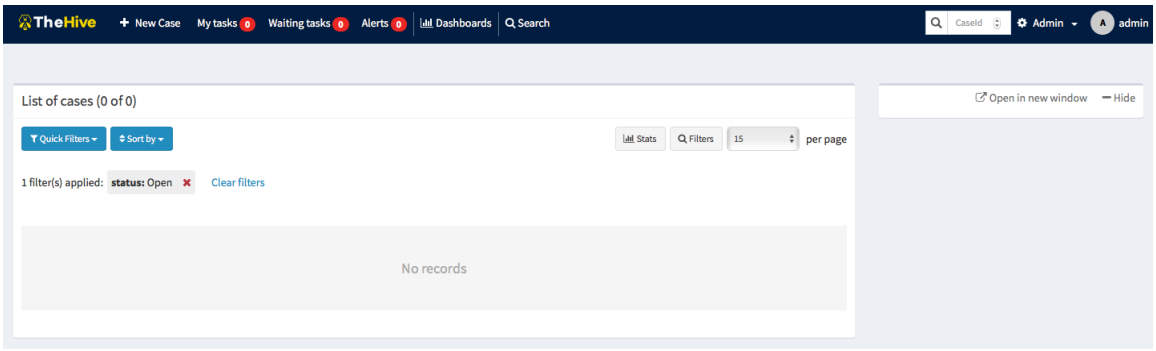
My local IP address is **192.168.43.188**. Then to enter TheHive I need to use this URL: **192.168.43.188:9000**

To access the platform use these credentials:

- **Login:** admin
- **Password:** thehive1234



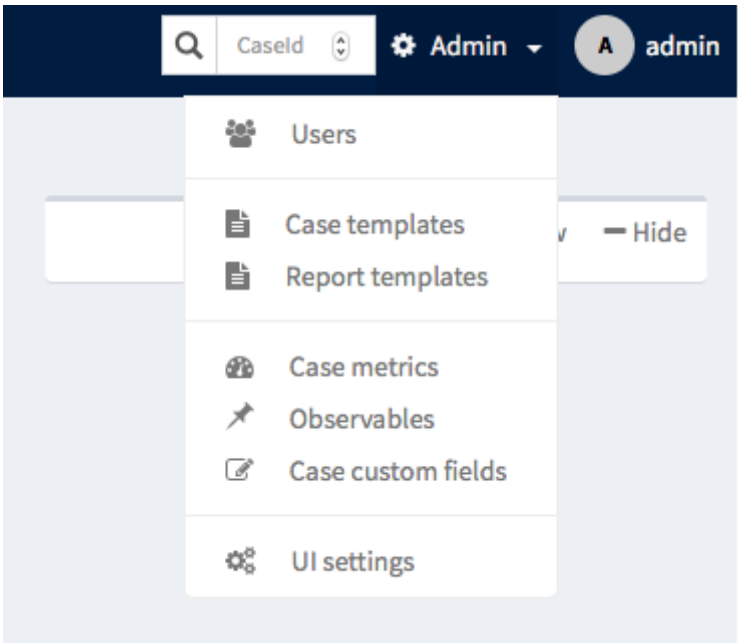
Voila! You are in the main dashboard



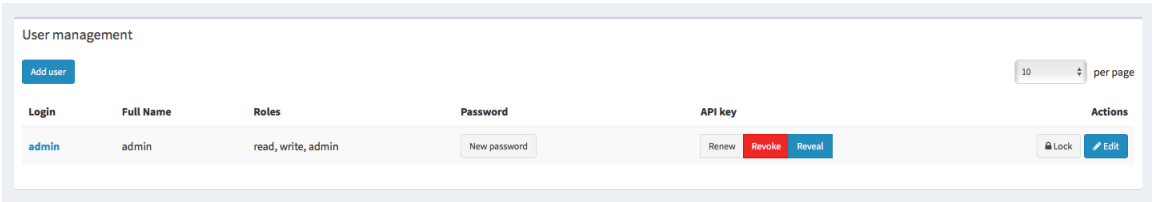
Let's start exploring how to use TheHive.

Users

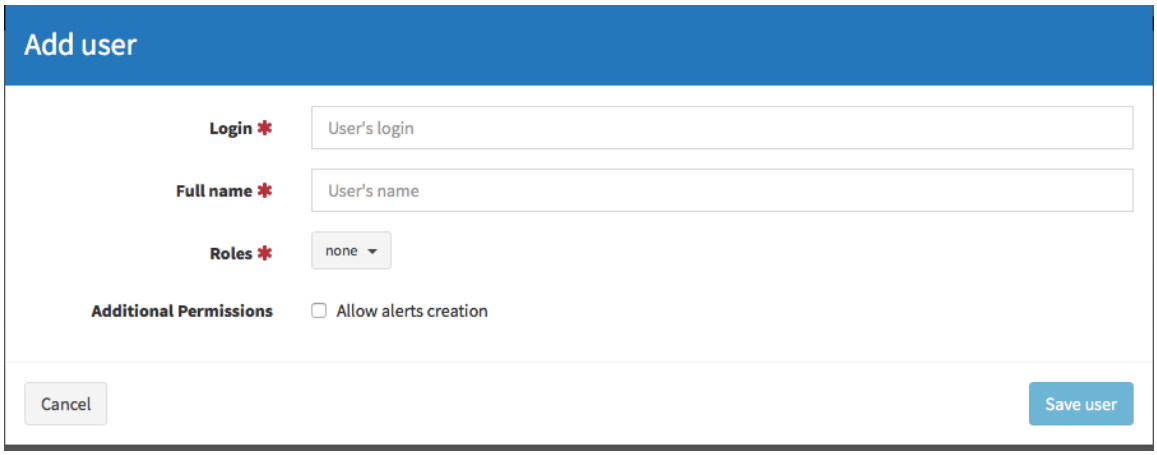
To create add your team members you need to create users. To create a user go to **Admin -\> Users** :



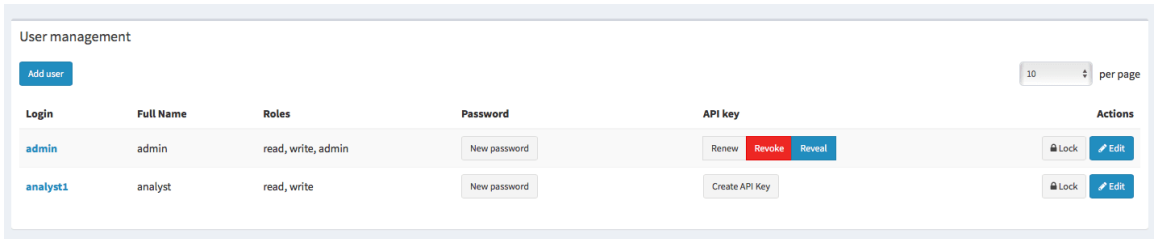
Click on "Add user"



Add your user information



The user was added successfully

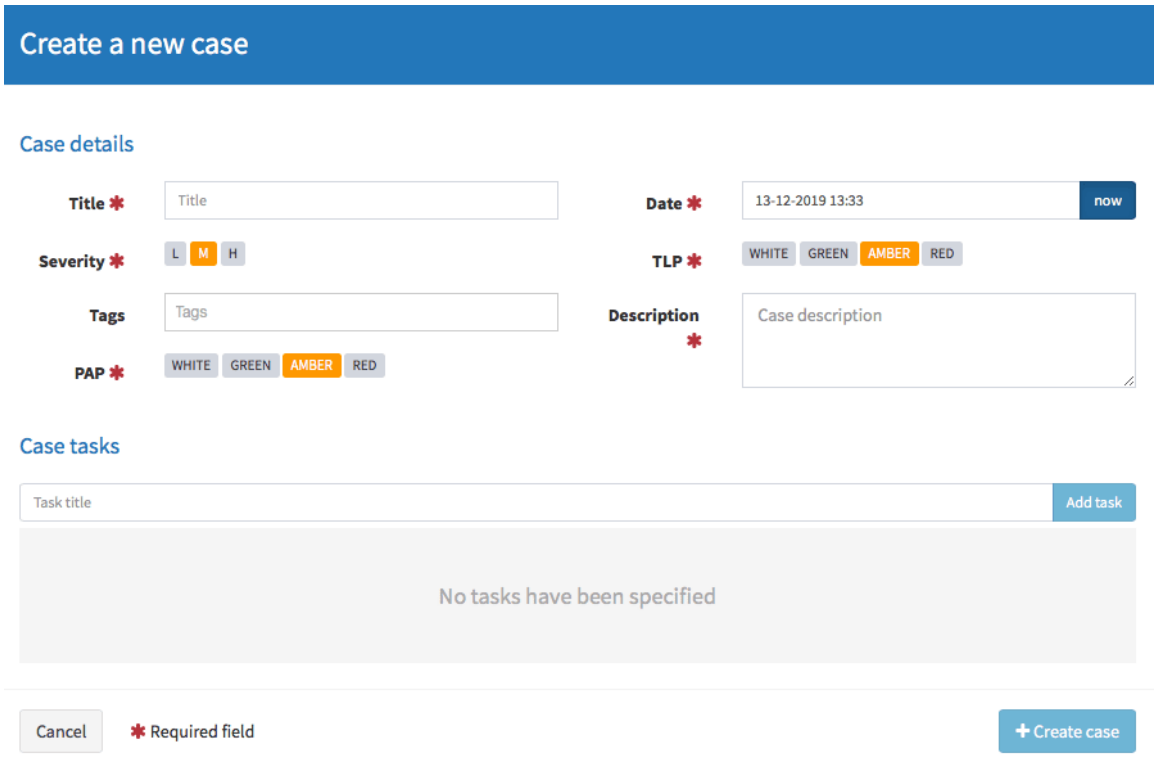


Create a new password for it by clicking " **New password**", type a password and press enter to save it.

Our password will be " **analyst1**" too.

Cases:

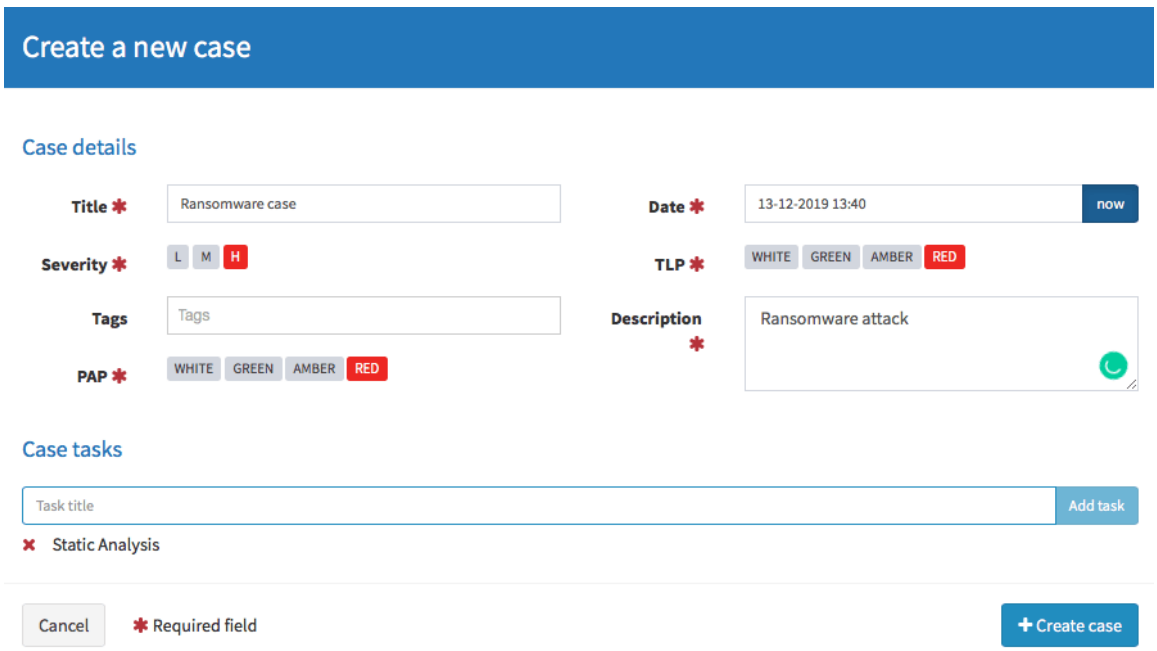
To create cases in the Hive, click on " **New case**"



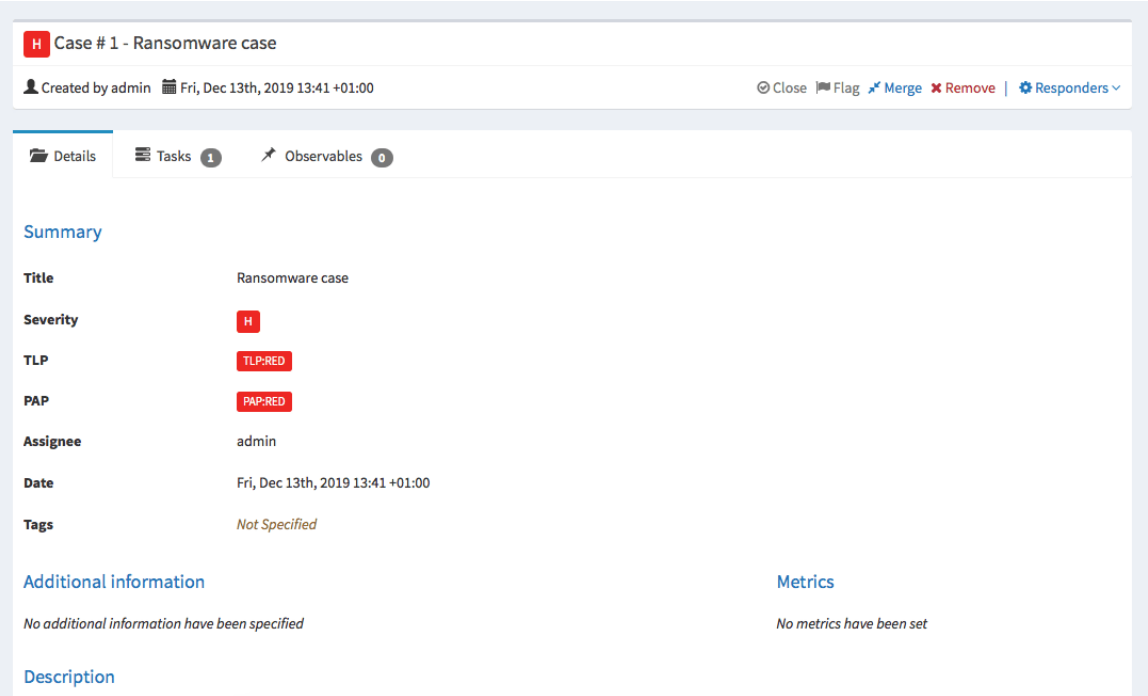
Add your case information:

- *Title*
- *Severity: Low, Medium or High*
- *Date*
- _Tags and so on. _

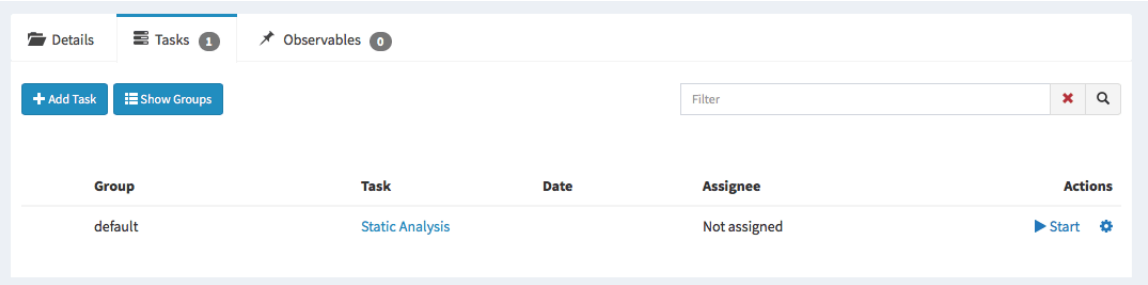
Add the case tasks:



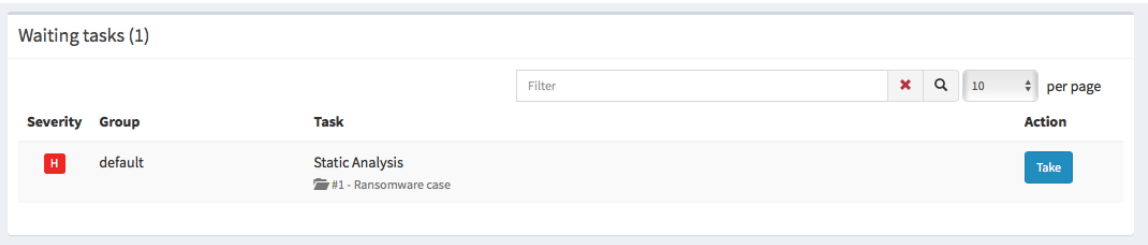
Now we created a case file



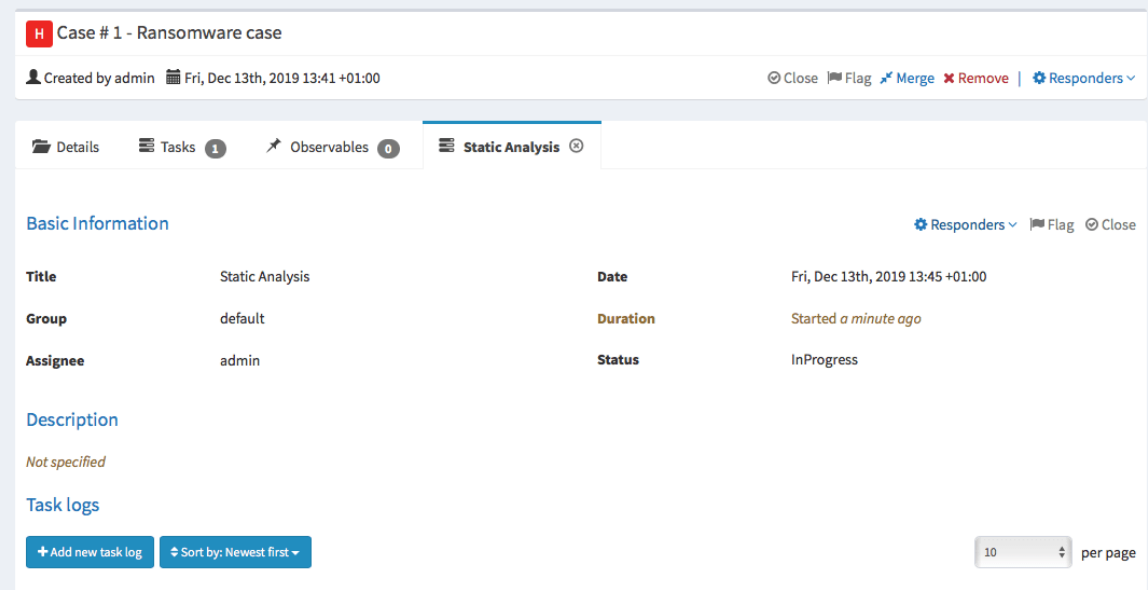
The case file contains also the tasks and the Observables:



You will find the case in the "Waiting cases" section



To take it just click on tasks and it will be added to your "my tasks" section

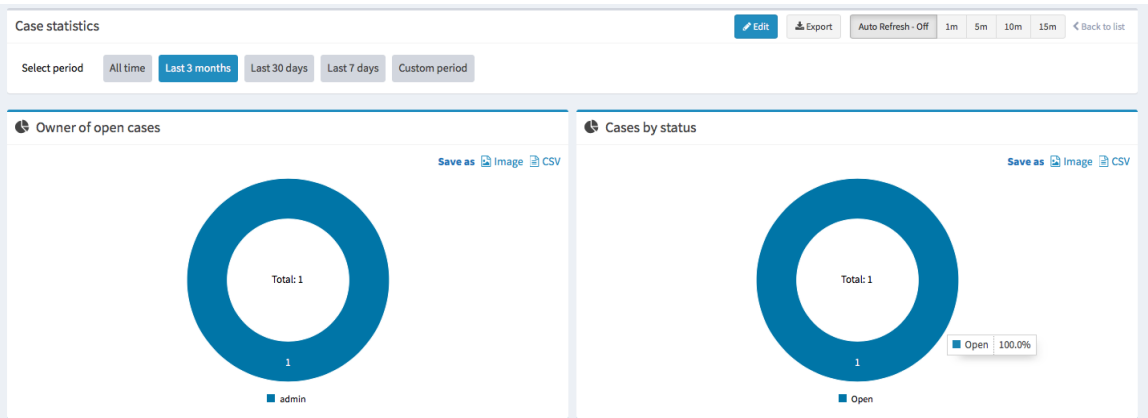


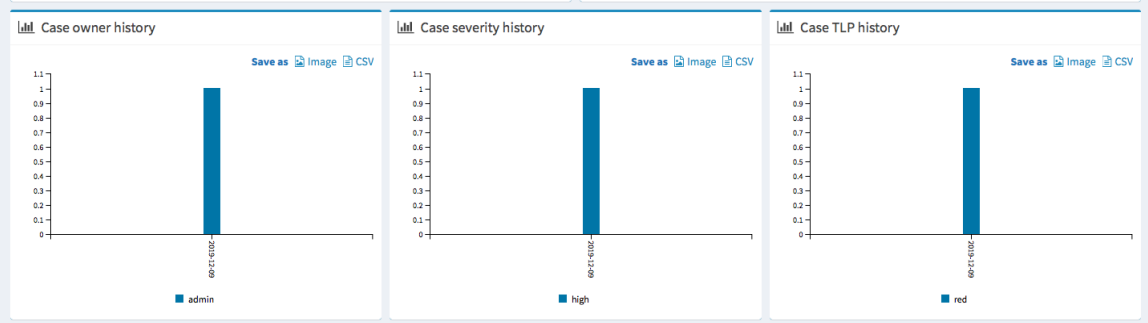
Once you finish the case, click on "Close" and it will be closed

Dashboards

To visualize your cases statistics you need to use The Hive dashboards. To open or create a new dashboard go to "Dashboards"

Select any available dashboard to explore it



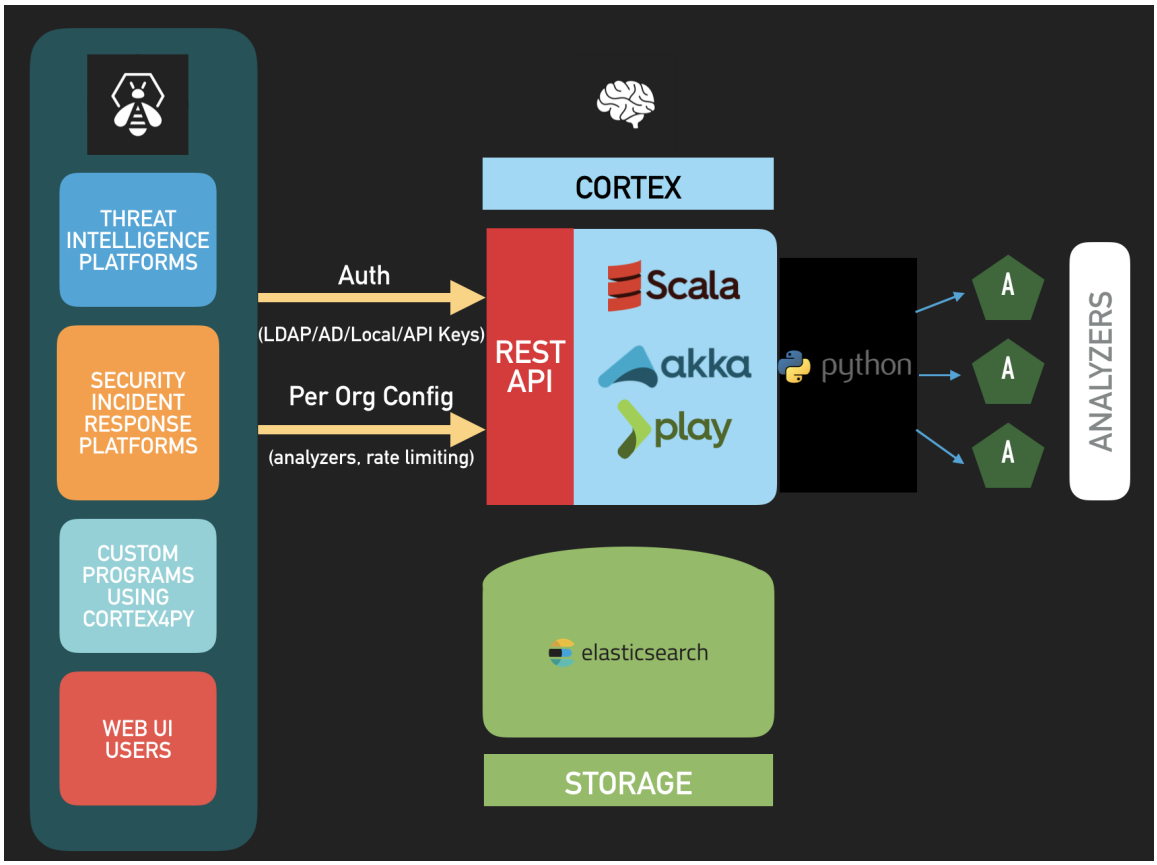


Cortex:

Its developers define cortex as follows:

> "Thanks to Cortex, observables such as IP and email addresses, URLs, domain names, files or hashes can be analyzed using a Web interface. Analysts can also automate these operations and submit large sets of observables from TheHive or through the Cortex REST API from alternative SIRP platforms, custom scripts or MISP. When used in conjunction with TheHive, Cortex largely facilitates the containment phase thanks to its Active Response features."

The following graph illustrates Cortex architecture:

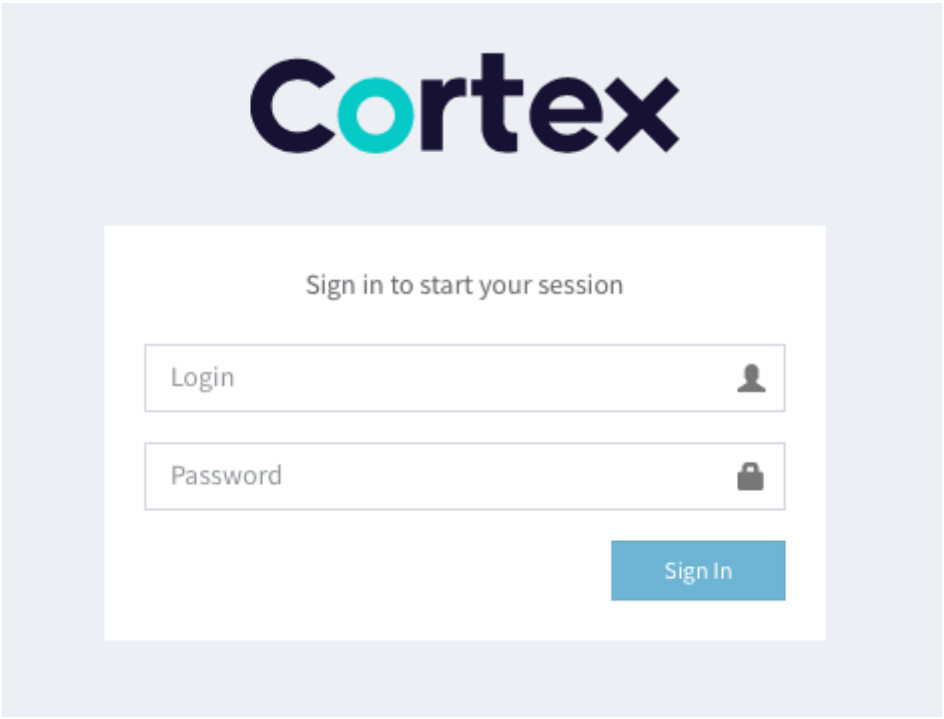


Figure

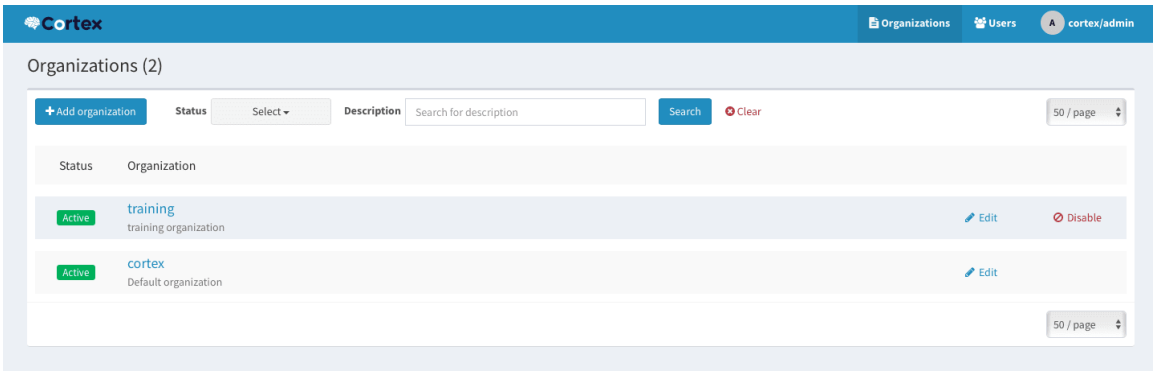
To enter cortex type this address on your browser: http://YOUR_SERVER_ADDRESS:9001/

Login to cortex using the same credentials as The hive

- Login: admin
- Password: thehive1234



This is the main dashboard of "Cortex"



Summary

In this guide, we discovered a great incident management platform called "the Hive" where we saw how to install it and use it to manage your team cases.

References:

- Recommendations of the National Institute of Standards and Technology: Computer Security Incident Handling Guide: https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-61r2.pdf
- Computer Security Incident Response Team (CSIRT) : http://whatis.techtarget.com/definition/Computer-Security-Incident-Response-Team-CSIRT

- US-CERT | United States Computer Emergency Readiness Team : https://www.us-cert.gov/about-us

Incident Response and Threat hunting with OSQuery and Fleet

In this guide, we are going to explore some powerful tools to help you enhance your incident response and threat hunting assessments. These tools are OSQuery and Kolide Fleet.

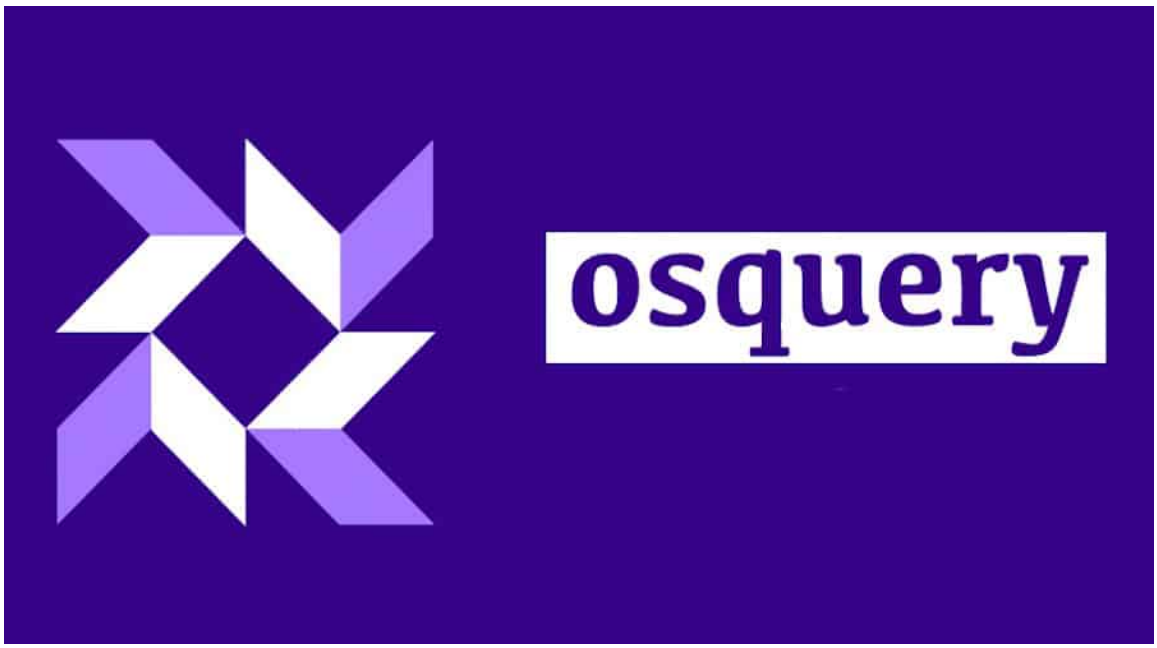
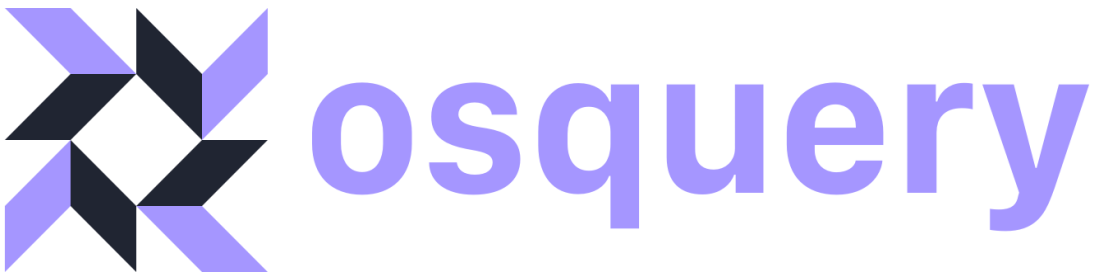


Image source: OSQUERY logo

Let's start exploring the first tool OSQuery

OSQuery Overview

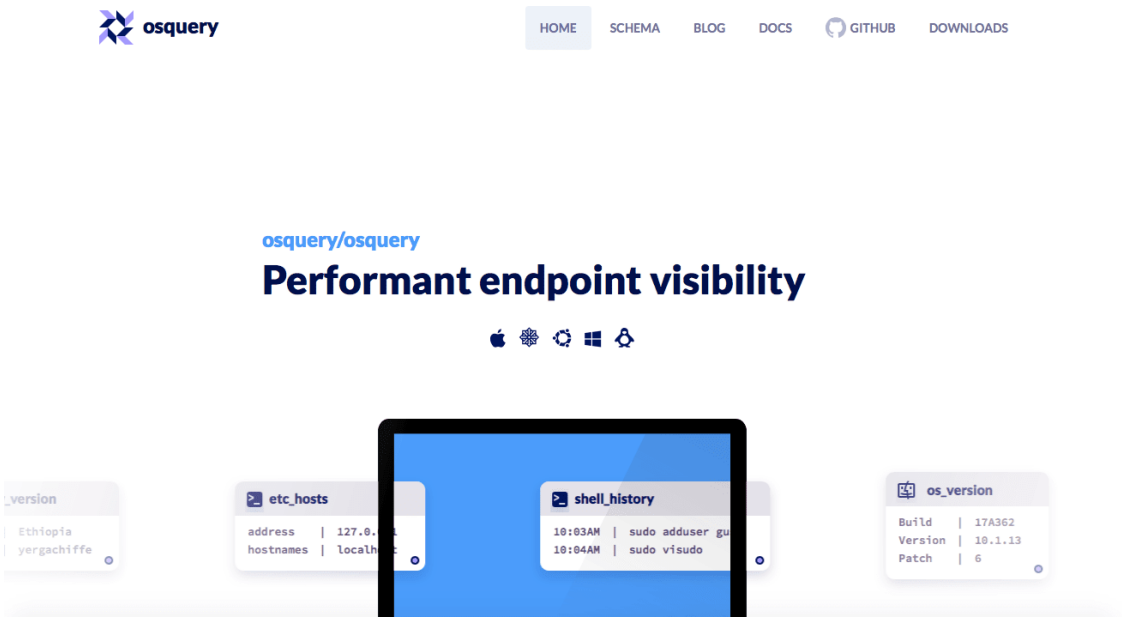
According to its official Github repository:



> *Osquery is a __ __ SQL __ __ powered __ __ operating system __ __ instrumentation, __ __ monitoring __, and__ __analytics__ __framework. It is Available for__ __Linux__ , __ __ macOS*

> __,__ __Windows,__and FreeBSD.

Its official website is https://osquery.io



To download OSQuery visit: https://osquery.io/downloads/official/4.3.0

Downloading & Installing Osquery

Packages and tarballs

These packages are built and signed by the osquery development team. They are mostly universal and use a minimal number of run-time library dependencies. This means the binaries are abnormally big (~20MB).

Osquery Version

4.3.0 (current)

Release Type

Official | Debug

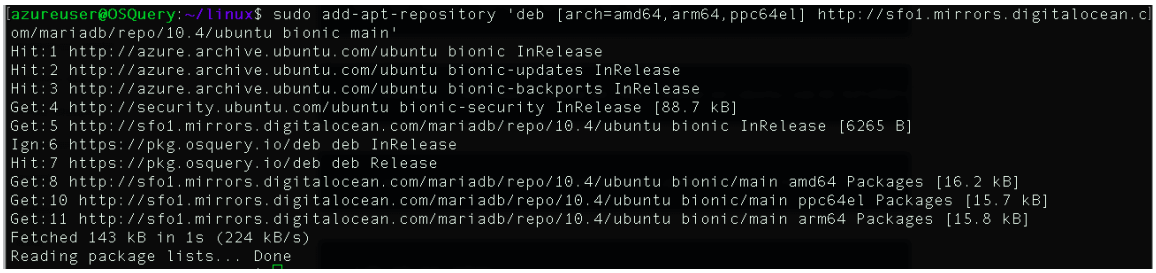
For the demonstration, we are going to use a Ubuntu 18.04 TLS server machine. To install it on our Ubuntu server type the following commands:

```
export OSQUERY\_KEY=1484120AC4E9F8A1A577AEEE97A80C63C9D8B80B
```

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys $OSQUERY\_KEY
```

```
azureuser@OSQuery:~$ export OSQUERY_KEY=1484120AC4E9F8A1A577AEEE97A80C63C9D8B80B
azureuser@OSQuery:~$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80
--recv-keys $OSQUERY_KEY
Executing: /tmp/apt-key-gpghome.Cfxz09rJve/gpg.1.sh --keyserver hkp://keyserver.
ubuntu.com:80 --recv-keys 1484120AC4E9F8A1A577AEEE97A80C63C9D8B80B
gpg: key 97A80C63C9D8B80B: public key "osquery (osquery) <osquery@fb.com>" impor
ted
gpg: Total number processed: 1
gpg:               imported: 1
```

```
sudo add-apt-repository &#39;deb [arch=amd64] [https://pkg.osquery.io/deb] (https://pkg.osquery.io/deb) deb main&#39;
```



`sudo apt-get update`

```
[azureuser@OSQuery:~$ sudo apt-get update
Hit:1 http://azure.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://azure.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB
]
Get:4 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:5 http://azure.archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [857
0 kB]
Get:6 http://azure.archive.ubuntu.com/ubuntu bionic/universe Translation-en [494
1 kB]
Get:7 http://azure.archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [1
51 kB]
Get:8 http://azure.archive.ubuntu.com/ubuntu bionic/multiverse Translation-en [1
08 kB]
Get:9 http://azure.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages
[969 kB]
Get:10 http://azure.archive.ubuntu.com/ubuntu bionic-updates/main Translation-en
 [329 kB]
Get:11 http://azure.archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Pa
ckages [60.5 kB]
Get:12 http://azure.archive.ubuntu.com/ubuntu bionic-updates/restricted Translat
ion-en [14.7 kB]
Get:13 http://azure.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Pack
ages [1081 kB]
```

`sudo apt-get install osquery`

```
[azureuser@OSQuery:~$ sudo apt-get install osquery
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  grub-pc-bin linux-headers-4.15.0-101
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  osquery
0 upgraded, 1 newly installed, 0 to remove and 29 not upgraded.
Need to get 9990 kB of archives.
After this operation, 56.5 MB of additional disk space will be used.
Get:1 https://pkg.osquery.io/deb deb/main amd64 osquery amd64 4.3.0-1.linux [999
0 kB]
Fetched 9990 kB in 1s (13.0 MB/s)
Selecting previously unselected package osquery.
(Reading database ... 76606 files and directories currently installed.)
Preparing to unpack .../osquery_4.3.0-1.linux_amd64.deb ...
Unpacking osquery (4.3.0-1.linux) ...
Setting up osquery (4.3.0-1.linux) ...
1797
Processing triggers for systemd (237-3ubuntu10.40) ...
Processing triggers for ureadahead (0.100.0-21) ...
```

OSQuery delivers these modes:

- **Osqueryi:** Interactive shell
- **Osqueryd:** Deamon

To start using OSQuery simply type:

```
osqueryi
```

To explore the available commands type **.help**

```
azureuser@OSQuery:~$ osqueryi
Using a virtual database. Need help, type '.help'
osquery> .help
Welcome to the osquery shell. Please explore your OS!
You are connected to a transient 'in-memory' virtual database.

.all [TABLE]     Select all from a table
.bail ON|OFF     Stop after hitting an error
.echo ON|OFF     Turn command echo on or off
.exit            Exit this program
.features        List osquery's features and their statuses
.headers ON|OFF  Turn display of headers on or off
.help            Show this message
.mode MODE       Set output mode where MODE is one of:
                   csv      Comma-separated values
                   column   Left-aligned columns see .width
                   line     One value per line
                   list     Values delimited by .separator string
                   pretty   Pretty printed SQL results (default)
.nullvalue STR   Use STRING in place of NULL values
.print STR...    Print literal STRING
.quit            Exit this program
.schema [TABLE]  Show the CREATE statements
```

To explore the available tables type

```
.tables
```

```
osquery> .tables
  => acpi_tables
  => apparmor_profiles
  => apt_sources
  => arp_cache
  => atom_packages
  => augeas
  => authorized_keys
  => block_devices
  => carbon_black_info
  => carves
  => chrome_extension_content_scripts
  => chrome_extensions
  => cpu_time
  => cpuid
  => crontab
  => curl
  => curl_certificate
  => deb_packages
  => device_file
  => device_hash
  => device_partitions
  => disk_encryption
  => dns_resolvers
```

To explore the schema of a specific table type

```
.schema <TABLE_HERE>
```

```
osquery> .schema yara
CREATE TABLE yara(`path` TEXT, `matches` TEXT, `count` INTEGER, `sig_group` TEXT, `sigfile` TE
XT, `strings` TEXT, `tags` TEXT, PRIMARY KEY (`path`, `sig_group`, `sigfile`)) WITHOUT ROWID;
CREATE TABLE yara_events(`target_path` TEXT, `category` TEXT, `action` TEXT, `transaction_id`
BIGINT, `matches` TEXT, `count` INTEGER, `strings` TEXT, `tags` TEXT, `time` BIGINT, `eid` TEX
T HIDDEN);
```

For example if you want to get the users type:

```
select * from users ;
```

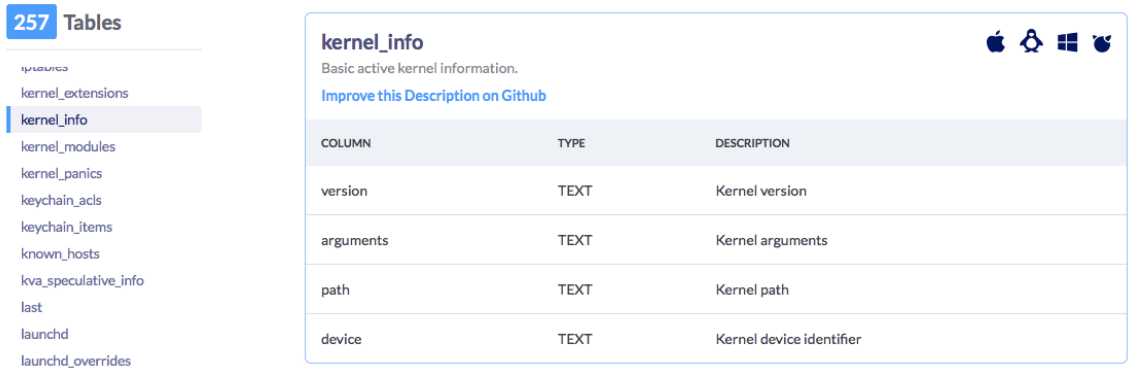
```
osquery> SELECT * FROM users;
+-------+-------+-----------+------------+----------------+-----------------------------------+
| uid   | gid   | uid_signed | gid_signed | username       | description                       |
+-------+-------+-----------+------------+----------------+-----------------------------------+
| 0     | 0     | 0          | 0          | root           | root                              |
| 1     | 1     | 1          | 1          | daemon         | daemon                            |
| 2     | 2     | 2          | 2          | bin            | bin                               |
| 3     | 3     | 3          | 3          | sys            | sys                               |
| 4     | 65534 | 4          | 65534      | sync           | sync                              |
| 5     | 60    | 5          | 60         | games          | games                             |
| 6     | 12    | 6          | 12         | man            | man                               |
| 7     | 7     | 7          | 7          | lp             | lp                                |
| 8     | 8     | 8          | 8          | mail           | mail                              |
| 9     | 9     | 9          | 9          | news           | news                              |
| 10    | 10    | 10         | 10         | uucp           | uucp                              |
| 13    | 13    | 13         | 13         | proxy          | proxy                             |
| 33    | 33    | 33         | 33         | www-data       | www-data                          |
| 34    | 34    | 34         | 34         | backup         | backup                            |
```

To select loggedin users type:

```
select * from logged_in_users ;
```

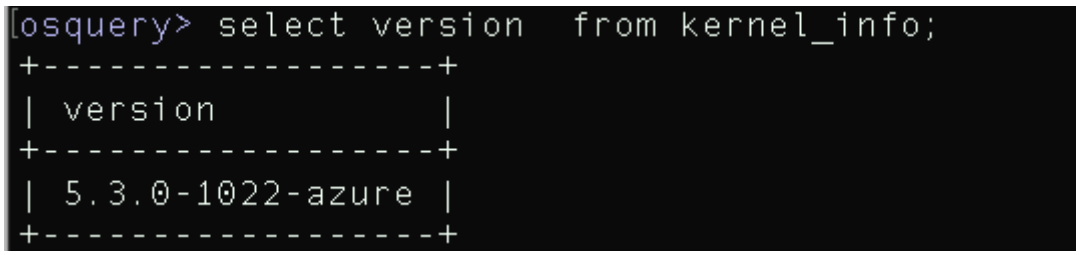
```
osquery> select * from logged_in_users ;
+-----------+-----------+-------+------------------+------------+------+
| type      | user      | tty   | host             | time       | pid  |
+-----------+-----------+-------+------------------+------------+------+
| boot_time | reboot    | ~     | 5.3.0-1022-azure | 1591772078 | 0    |
| login     | LOGIN     | tty1  |                  | 1591772117 | 1244 |
| login     | LOGIN     | ttyS0 |                  | 1591772117 | 1238 |
| runlevel  | runlevel  | ~     | 5.3.0-1022-azure | 1591772120 | 53   |
| user      | azureuser | pts/0 | 196.184.163.64   | 1591772172 | 1779 |
+-----------+-----------+-------+------------------+------------+------+
```

The official website contains the list of all the available tables and its schemes. For example this is the scheme of **Kernel_info table**



For example to select the version of the kernel type:

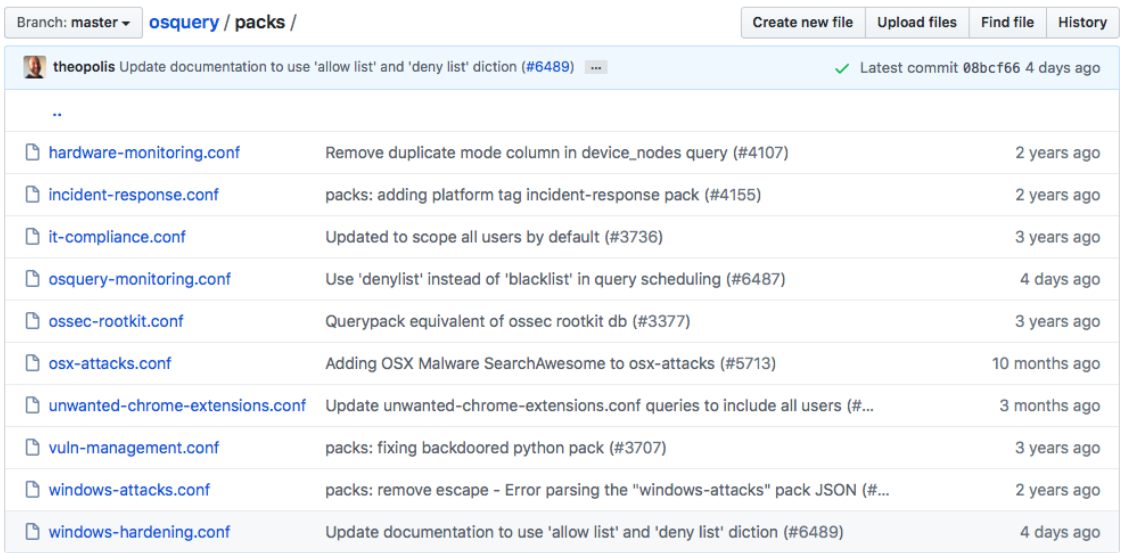
```
select version from Kernel_info
```



Let's suppose that you want to automate a specific query (selecting users) every 300 seconds. Edit the **/etc/osquery/osquery.conf** file and add your rules

"schedule": { "Users": { "query": "SELECT * FROM users;", "interval": 300 } },

A collection of queries is called a **Pack**. OSQuery provides many hekpful packs that you can use in your assessments here: https://github.com/osquery/osquery/tree/master/packs



This is a query from https://github.com/osquery/osquery/blob/master/packs/incident-response.conf that retreive all the startup items in MacOS hosts:

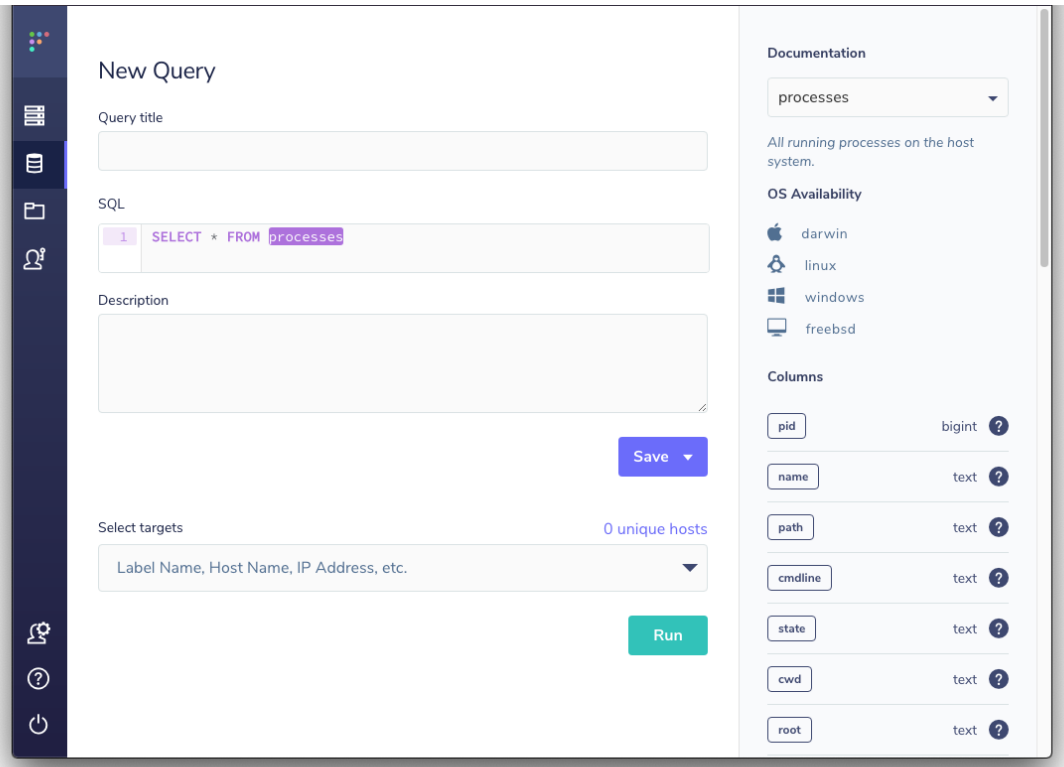
```
"startup_items": {
  "query" : "select * from startup_items;",
  "interval" : "86400",
  "platform" : "darwin",
  "version" : "1.4.5",
  "description" : "Retrieve all the items that will load when the target OSX system starts.",
  "value" : "Identify malware that uses this persistence mechanism to launch at a given interval"
},
```

But now, what to do if we want to deploy OSQuery in large scale environments and we want to manage them all easily. In this situation we need another powerful platform called "Kolide Fleet"

Kolide Fleet (OSQuery Management)

:heavy_exclamation_mark: Kolide is no longer maintaining Fleet. The new name is Fleet and can be found here: https://github.com/fleetdm/fleet

> Fleet is the most widely used open source osquery manager. Deploying osquery with Fleet enables programmable live queries, streaming logs, and effective management of osquery across 50,000+ servers, containers, and laptops. It's especially useful for talking to multiple devices at the same time.



According to its official Github repository:

> *Fleet is the most widely used __ __ open-source __ __ osquery Fleet manager. Deploying osquery with Fleet enables live queries, and effective __ __ management __ __ of osquery infrastructure.*

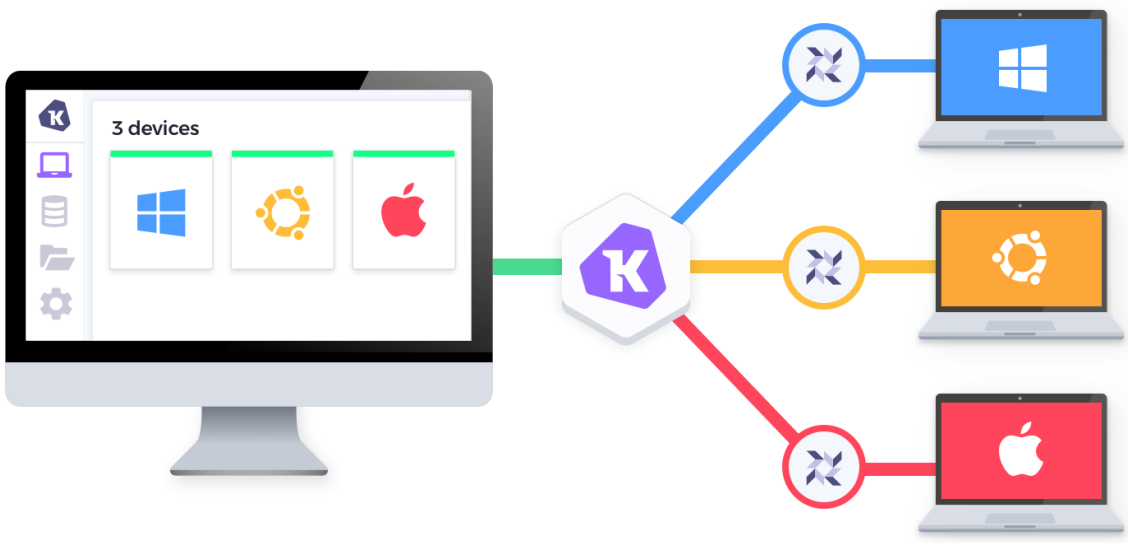


Image source: Kolide fleet

To install it use the following commands:

```
wget https://github.com/kolide/fleet/releases/latest/download/fleet.zip
```

```
azureuser@OSQuery:~$ wget https://github.com/kolide/fleet/releases/latest/download/fleet.zip
--2020-06-10 07:27:36--  https://github.com/kolide/fleet/releases/latest/download/fleet.zip
Resolving github.com (github.com)... 140.82.118.3
Connecting to github.com (github.com)|140.82.118.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github.com/kolide/fleet/releases/download/2.6.0/fleet.zip [following]
--2020-06-10 07:27:36--  https://github.com/kolide/fleet/releases/download/2.6.0/fleet.zip
Reusing existing connection to github.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/64099814/54aa1a00-
6dba-11ea-9324-6a129cdd3148?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CS
VEH53A%2F20200610%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200610T072737Z&X-Amz-Expires=
300&X-Amz-Signature=1750ac9a1ff866c2cc30c51c5d08e50e6a51169d6e61acfcea1a514b61427e0f&X-Amz-S
ignedHeaders=host&actor_id=0&repo_id=64099814&response-content-disposition=attachment%3B%20f
ilename%3Dfleet.zip&response-content-type=application%2Foctet-stream [following]
--2020-06-10 07:27:37--  https://github-production-release-asset-2e65be.s3.amazonaws.com/640
99814/54aa1a00-6dba-11ea-9324-6a129cdd3148?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential
=AKIAIWNJYAX4CSVEH53A%2F20200610%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200610T072737Z
&X-Amz-Expires=300&X-Amz-Signature=1750ac9a1ff866c2cc30c51c5d08e50e6a51169d6e61acfcea1a514b6
1427e0f&X-Amz-SignedHeaders=host&actor_id=0&repo_id=64099814&response-content-disposition=at
tachment%3B%20filename%3Dfleet.zip&response-content-type=application%2Foctet-stream
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release
-asset-2e65be.s3.amazonaws.com)... 52.216.230.139
Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-production-rel
```

```
sudo apt-get install unzip
```

Unzip the file:

```
sudo unzip fleet.zip
```

```
azureuser@OSQuery:~$ sudo unzip fleet.zip
Archive:  fleet.zip
   creating: darwin/
  inflating: darwin/fleetctl
  inflating: darwin/fleet
   creating: linux/
  inflating: linux/fleetctl
  inflating: linux/fleet
   creating: windows/
  inflating: windows/fleetctl.exe
  inflating: windows/fleet.exe
```

Enter the linux folder:

```
azureuser@OSQuery:~$ ls
darwin  fleet.zip  linux  windows
azureuser@OSQuery:~$ cd linux
azureuser@OSQuery:~/linux$ ls
fleet  fleetctl
azureuser@OSQuery:~/linux$
```

Copy the binaries in /usr/bin

```
sudo cp * /usr/bin/
```

Install this required program:

```
sudo apt install software-properties-common
```

```
azureuser@OSQuery:~/linux$ sudo apt install software-properties-common
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  grub-pc-bin linux-headers-4.15.0-101
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  python3-software-properties
The following packages will be upgraded:
  python3-software-properties software-properties-common
2 upgraded, 0 newly installed, 0 to remove and 27 not upgraded.
Need to get 33.8 kB of archives.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://azure.archive.ubuntu.com/ubuntu bionic-updates/main amd64 software-properties-c
ommon all 0.96.24.32.13 [10.0 kB]
Get:2 http://azure.archive.ubuntu.com/ubuntu bionic-updates/main amd64 python3-software-prop
erties all 0.96.24.32.13 [23.8 kB]
Fetched 33.8 kB in 0s (1406 kB/s)
(Reading database ... 76870 files and directories currently installed.)
Preparing to unpack .../software-properties-common_0.96.24.32.13_all.deb ...
Unpacking software-properties-common (0.96.24.32.13) over (0.96.24.32.12) ...
Preparing to unpack .../python3-software-properties_0.96.24.32.13_all.deb ...
```

```
sudo apt-key adv --recv-keys --keyserver hkp://keyserver.ubuntu.com:80
0xF1656F24C74CD1D8
```

```
add-apt-repository 'deb [arch=amd64,arm64,ppc64el]
http://sfo1.mirrors.digitalocean.com/mariadb/repo/10.4/ubuntu bionic main'
```

```
azureuser@OSQuery:~/linux$ sudo add-apt-repository 'deb [arch=amd64,arm64,ppc64el] http://sfo1.mirrors.digitalocean.c
om/mariadb/repo/10.4/ubuntu bionic main'
Hit:1 http://azure.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://azure.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://azure.archive.ubuntu.com/ubuntu bionic-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:5 http://sfo1.mirrors.digitalocean.com/mariadb/repo/10.4/ubuntu bionic InRelease [6265 B]
Ign:6 https://pkg.osquery.io/deb deb InRelease
Hit:7 https://pkg.osquery.io/deb deb Release
Get:8 http://sfo1.mirrors.digitalocean.com/mariadb/repo/10.4/ubuntu bionic/main amd64 Packages [16.2 kB]
Get:10 http://sfo1.mirrors.digitalocean.com/mariadb/repo/10.4/ubuntu bionic/main ppc64el Packages [15.7 kB]
Get:11 http://sfo1.mirrors.digitalocean.com/mariadb/repo/10.4/ubuntu bionic/main arm64 Packages [15.8 kB]
Fetched 143 kB in 1s (224 kB/s)
Reading package lists... Done
```

```
sudo apt-get update
```

```
azureuser@OSQuery:~$ sudo apt-get update
Hit:1 http://azure.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://azure.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://azure.archive.ubuntu.com/ubuntu bionic-backports InRelease
Ign:4 https://pkg.osquery.io/deb deb InRelease
Hit:5 https://pkg.osquery.io/deb deb Release
Get:6 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Fetched 88.7 kB in 1s (158 kB/s)
Reading package lists... Done
```

Install Maria database server and its client:

```
sudo apt install mariadb-server mariadb-client
```

```
azureuser@OSQuery:~/linux$ sudo apt install mariadb-server mariadb-client
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  grub-pc-bin linux-headers-4.15.0-101
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  galera-4 libaio1 libcgi-fast-perl libcgi-pm-perl libdbd-mysql-perl libdbi-perl libencode-locale-perl libfcgi-perl
  libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl libhttp-date-perl libhttp-message-perl
  libio-html-perl liblwp-mediatypes-perl libmariadb3 libmysqlclient20 libterm-readkey-perl libtimedate-perl
  liburi-perl mariadb-client-10.4 mariadb-client-core-10.4 mariadb-common mariadb-server-10.4
  mariadb-server-core-10.4 mysql-common socat
Suggested packages:
  libclone-perl libmldbm-perl libnet-daemon-perl libsql-statement-perl libdata-dump-perl libipc-sharedcache-perl
  libwww-perl mailx mariadb-test tinyca
The following NEW packages will be installed:
  galera-4 libaio1 libcgi-fast-perl libcgi-pm-perl libdbd-mysql-perl libdbi-perl libencode-locale-perl libfcgi-perl
  libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl libhttp-date-perl libhttp-message-perl
  libio-html-perl liblwp-mediatypes-perl libmariadb3 libmysqlclient20 libterm-readkey-perl libtimedate-perl
  liburi-perl mariadb-client mariadb-client-10.4 mariadb-client-core-10.4 mariadb-common mariadb-server
  mariadb-server-10.4 mariadb-server-core-10.4 mysql-common socat
0 upgraded, 29 newly installed, 0 to remove and 27 not upgraded.
Need to get 25.0 MB of archives.
```

Check its status:

`sudo systemctl status mariadb`

```
azureuser@OSQuery:~/linux$ sudo systemctl status mariadb
● mariadb.service - MariaDB 10.4.13 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset: enabled)
  Drop-In: /etc/systemd/system/mariadb.service.d
           └─migrated-from-my.cnf-settings.conf
   Active: active (running) since Wed 2020-06-10 07:36:20 UTC; 32s ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
 Main PID: 19331 (mysqld)
   Status: "Taking your SQL requests now..."
    Tasks: 32 (limit: 1024)
   CGroup: /system.slice/mariadb.service
           └─19331 /usr/sbin/mysqld

Jun 10 07:36:29 OSQuery /etc/mysql/debian-start[19389]: information_schema
Jun 10 07:36:29 OSQuery /etc/mysql/debian-start[19389]: mysql
Jun 10 07:36:29 OSQuery /etc/mysql/debian-start[19389]: performance_schema
Jun 10 07:36:29 OSQuery /etc/mysql/debian-start[19389]: Phase 6/7: Checking and upgrading tables
Jun 10 07:36:29 OSQuery /etc/mysql/debian-start[19389]: Processing databases
Jun 10 07:36:29 OSQuery /etc/mysql/debian-start[19389]: information_schema
Jun 10 07:36:29 OSQuery /etc/mysql/debian-start[19389]: performance_schema
Jun 10 07:36:29 OSQuery /etc/mysql/debian-start[19389]: Phase 7/7: Running 'FLUSH PRIVILEGES'
Jun 10 07:36:29 OSQuery /etc/mysql/debian-start[19389]: OK
Jun 10 07:36:29 OSQuery /etc/mysql/debian-start[20187]: Triggering myisam-recover for all MyISAM tables and aria-reco
```

Enable Mariadb service:

`sudo systemctl is-enabled mariadb`

```
azureuser@OSQuery:~/linux$ sudo systemctl is-enabled mariadb
enabled
```

Enter mysql and type the following commands:

`sudo mysql -u root -p`

```
azureuser@OSQuery:~/linux$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 49
Server version: 10.4.13-MariaDB-1:10.4.13+maria~bionic-log mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

`create database kolide;`

```
grant all on kolide.* to kolideuser@localhost identified by 'Passw0rd!';
```

```
MariaDB [(none)]> grant all on kolide.* to kolideuser@localhost identified by 'Passw0rd!';
Query OK, 0 rows affected (0.147 sec)
```

```
flush privileges;
```

```
exit
```

Install Redis:

```
sudo apt install redis
```

```
azureuser@OSQuery:~/linux$ sudo apt install redis
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  grub-pc-bin linux-headers-4.15.0-101
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libjemalloc1 redis-server redis-tools
Suggested packages:
  ruby-redis
The following NEW packages will be installed:
  libjemalloc1 redis redis-server redis-tools
0 upgraded, 4 newly installed, 0 to remove and 27 not upgraded.
Need to get 637 kB of archives.
After this operation, 3083 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://azure.archive.ubuntu.com/ubuntu bionic/universe amd64 libjemalloc1 amd64 3.6.0-11 [82.4 kB]
Get:2 http://azure.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 redis-tools amd64 5:4.0.9-1ubuntu0.2 [516
kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 redis-server amd64 5:4.0.9-1ubuntu0.2 [35.
4 kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 redis all 5:4.0.9-1ubuntu0.2 [3084 B]
Fetched 637 kB in 0s (12.4 MB/s)
```

Prepare fleet:

```
fleet prepare db --mysql_address=127.0.0.1:3306 --mysql_database=kolide --
mysql_username=kolideuser --mysql_password=Passw0rd!
```

```
fleet serve --mysql_address=127.0.0.1:3306 \
```

```
--mysql_database=kolide --mysql_username=kolideuser --mysql_password=Passw0rd! \
```

```
--server_cert=/etc/ssl/certs/kolide.cert --server_key=/etc/ssl/private/kolide.key
\
```

```
--logging_json
```

```
azureuser@OSQuery:~/linux$ sudo fleet serve --mysql_address=127.0.0.1:3306 \
> --mysql_database=kolide --mysql_username=kolideuser --mysql_password=Passw0rd! \
> --server_cert=/etc/ssl/certs/kolide.cert --server_key=/etc/ssl/private/kolide.key \
> --logging_json
################################################################################
# ERROR:
#   A value must be supplied for --auth_jwt_key. This value is used to create
#   session tokens for users.
#
#   Consider using the following randomly generated key:
#   9yKI2MeThUSLtsYiCS7etUSJZD1lgHLr
################################################################################
```

```
sudo fleet serve --mysql_address=127.0.0.1:3306 \
```

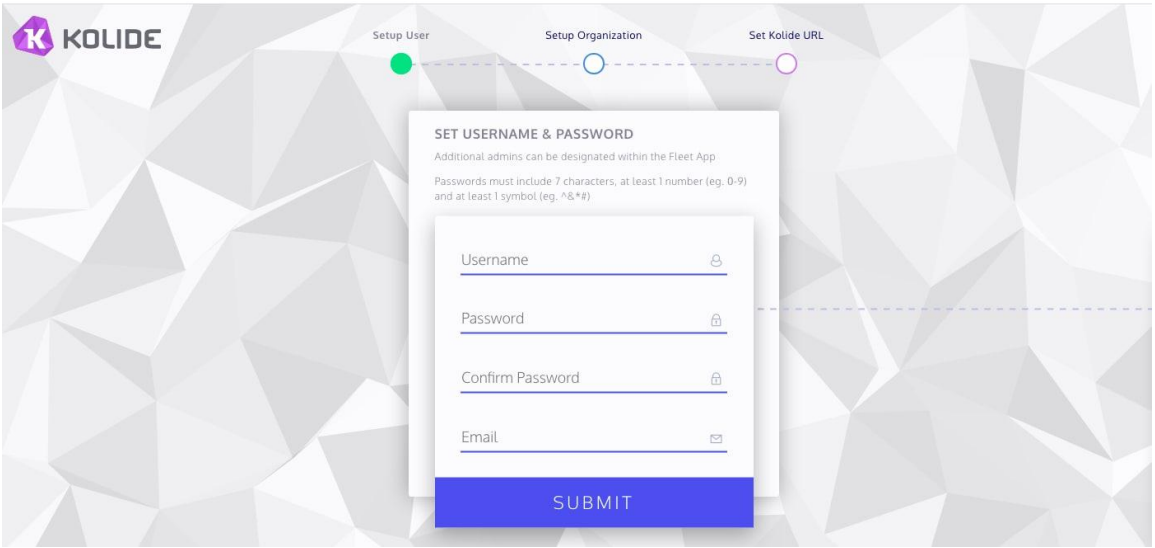
```
--mysql_database=kolide --mysql_username=kolideuser --mysql_password=Passw0rd!
\

--server_cert=/etc/ssl/certs/kolide.cert --
server_key=/etc/ssl/private/kolide.key \

--logging_json --auth_jwt_key=9yKI2MeThUSLtsYiCS7etUSJZD1lgHLr
```

Start fleet:

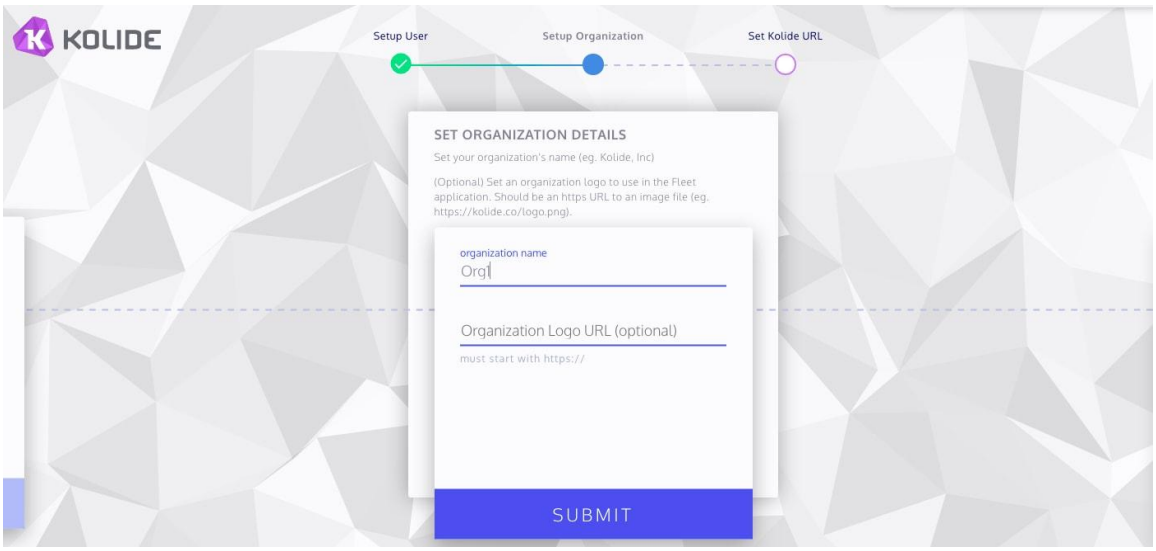
```
azureuser@OSQuery:~/linux$ sudo fleet serve --mysql_address=127.0.0.1:3306 \
> --mysql_database=kolide --mysql_username=kolideuser --mysql_password=Passw0rd! \
> --server_cert=/etc/ssl/certs/kolide.cert --server_key=/etc/ssl/private/kolide.key \
> --logging_json --auth_jwt_key=9yKI2MeThUSLtsYiCS7etUSJZD1lgHLr
{"component":"service","err":null,"method":"ListUsers","took":"756.204µs","ts":"2020-06-10T07:46:03.690481794Z","user
":"none"}
{"address":"0.0.0.0:8080","msg":"listening","transport":"https","ts":"2020-06-10T07:46:03.692549405Z"}
```

Go to https://\<SERVER_IP\>:8080

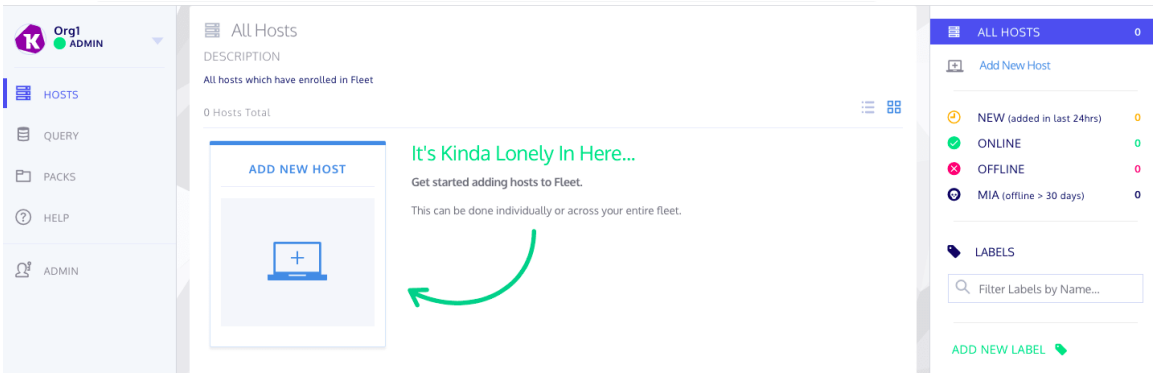
Provide your username, password and email



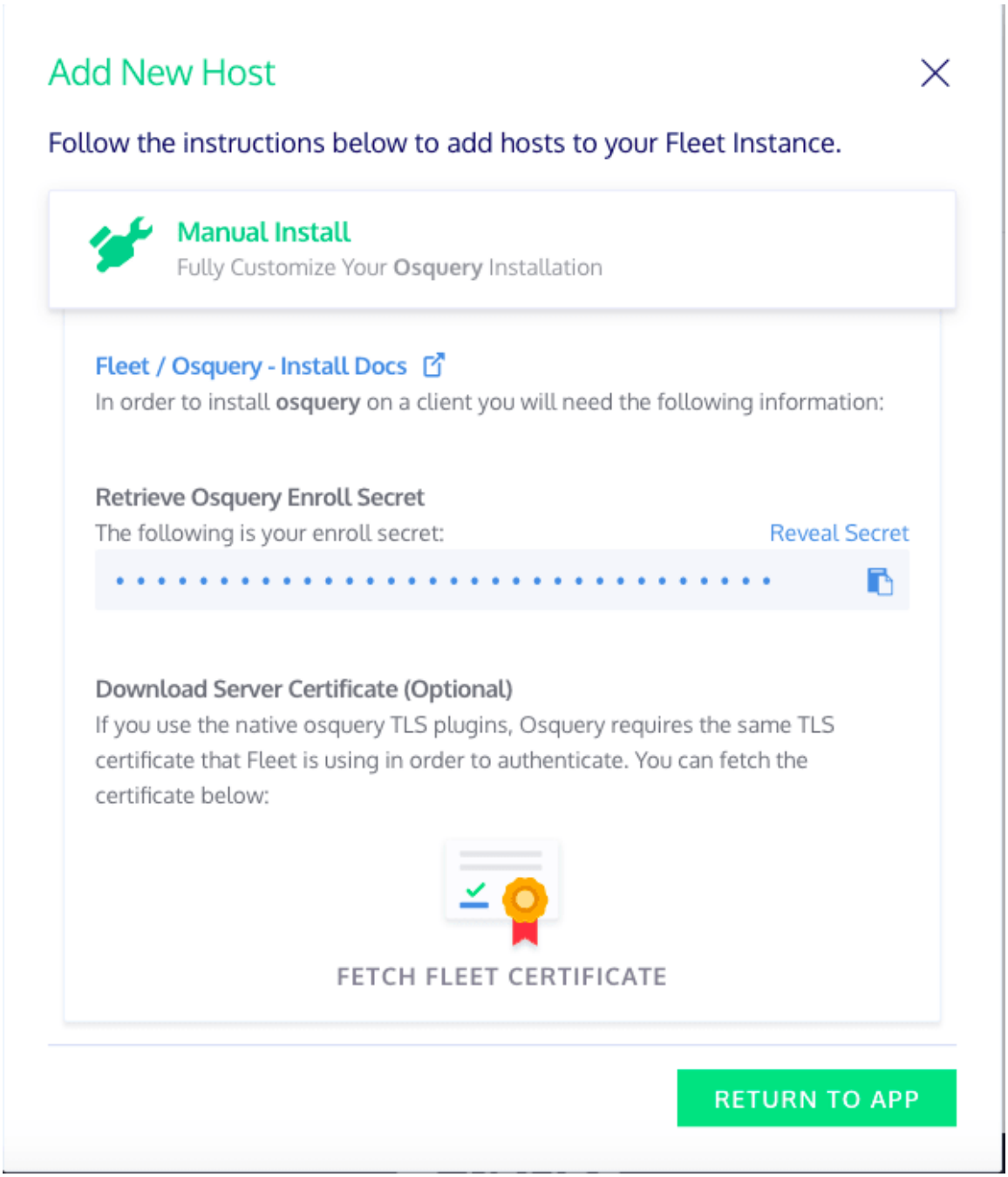
Add your organization name, the organization domain name/IP and submit:



Voila! Kolide fleet is deployed successfully.



Now let's add our host. To do so, click on "ADD NEW HOST" and you will get this window. It provides a key called "OSQuery enroll secret" that we are going to use later.



To add the host, we need to install the fleet launcher. In our case we are using the same host.

wget https://github.com/kolide/launcher/releases/download/v0.11.10/launcher_v0.11.10.zip

```
azureuser@OSQuery:~$ wget https://github.com/kolide/launcher/releases/download/v
0.11.10/launcher_v0.11.10.zip
--2020-06-10 07:53:56--  https://github.com/kolide/launcher/releases/download/v0
.11.10/launcher_v0.11.10.zip
Resolving github.com (github.com)... 140.82.118.3
Connecting to github.com (github.com)|140.82.118.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/900722
96/cb456c00-8633-11ea-8623-b43336277d08?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-C
redential=AKIAIWNJYAX4CSVEH53A%2F20200610%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-
Date=20200610T075356Z&X-Amz-Expires=300&X-Amz-Signature=78cd74f220b57758a3a9cc3f
f49a02eb35f8de44fee1956d665b9f1adb961738&X-Amz-SignedHeaders=host&actor_id=0&rep
o_id=90072296&response-content-disposition=attachment%3B%20filename%3Dlauncher_v
0.11.10.zip&response-content-type=application%2Foctet-stream [following]
--2020-06-10 07:53:56--  https://github-production-release-asset-2e65be.s3.amazo
naws.com/90072296/cb456c00-8633-11ea-8623-b43336277d08?X-Amz-Algorithm=AWS4-HMAC
-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20200610%2Fus-east-1%2Fs3%2Faws4
_request&X-Amz-Date=20200610T075356Z&X-Amz-Expires=300&X-Amz-Signature=78cd74f22
0b57758a3a9cc3ff49a02eb35f8de44fee1956d665b9f1adb961738&X-Amz-SignedHeaders=host
&actor_id=0&repo_id=90072296&response-content-disposition=attachment%3B%20filena
me%3Dlauncher_v0.11.10.zip&response-content-type=application%2Foctet-stream
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-produc
tion-release-asset-2e65be.s3.amazonaws.com)... 52.217.9.68
Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-pr
```

Unzip the file:

```
sudo unzip launcher\_v0.11.10.zip
```

```
azureuser@OSQuery:~$ sudo unzip launcher_v0.11.10.zip
Archive:  launcher_v0.11.10.zip
  inflating: darwin/launcher
  inflating: darwin/osquery-extension.ext
  inflating: darwin/osqueryd
  inflating: linux/launcher
  inflating: linux/osquery-extension.ext
  inflating: linux/osqueryd
  inflating: windows/osquery-extension.exe
  inflating: windows/launcher.exe
  inflating: windows/osqueryd.exe
azureuser@OSQuery:~$ cd linux
azureuser@OSQuery:~/linux$ ls
fleet  fleetctl  launcher  osquery-extension.ext  osqueryd
```

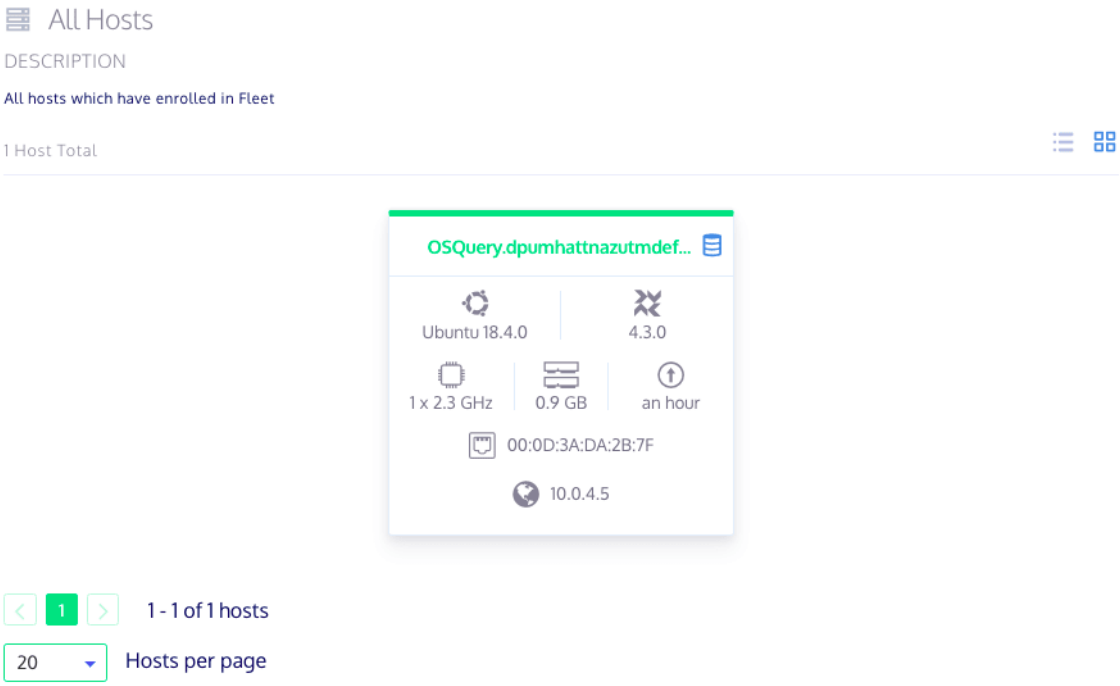
Enter the Linux file:

```
cd linux
```

Start the launcher

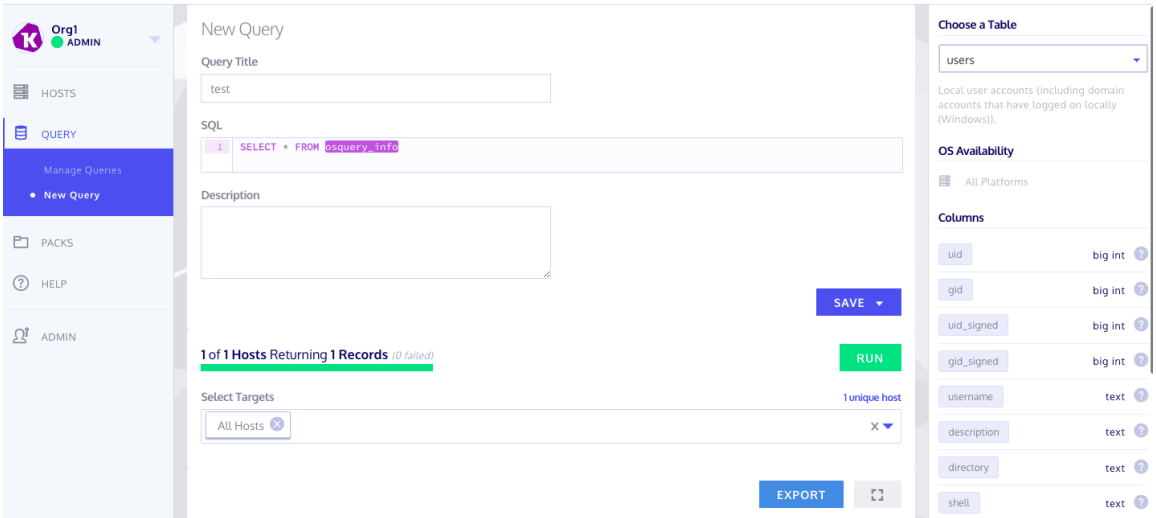
```
./launcher   --hostname=127.0.0.1:8080   --root_directory=$(mktemp -d)   --
enroll_secret=<COPY SECRET KEY HERE> --insecure
```

Congratulation! if you refresh the Kolide fleet dashboard you will see the newly added host

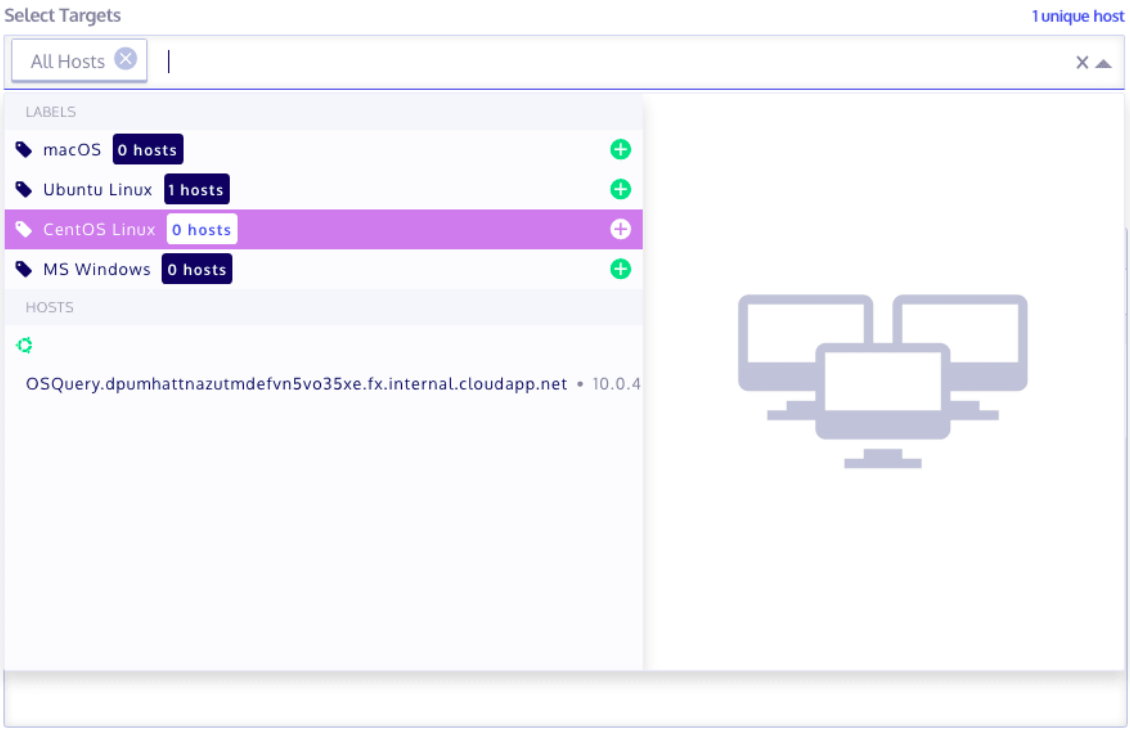


To run and add queries go to **QUERY** -\> **New Query**

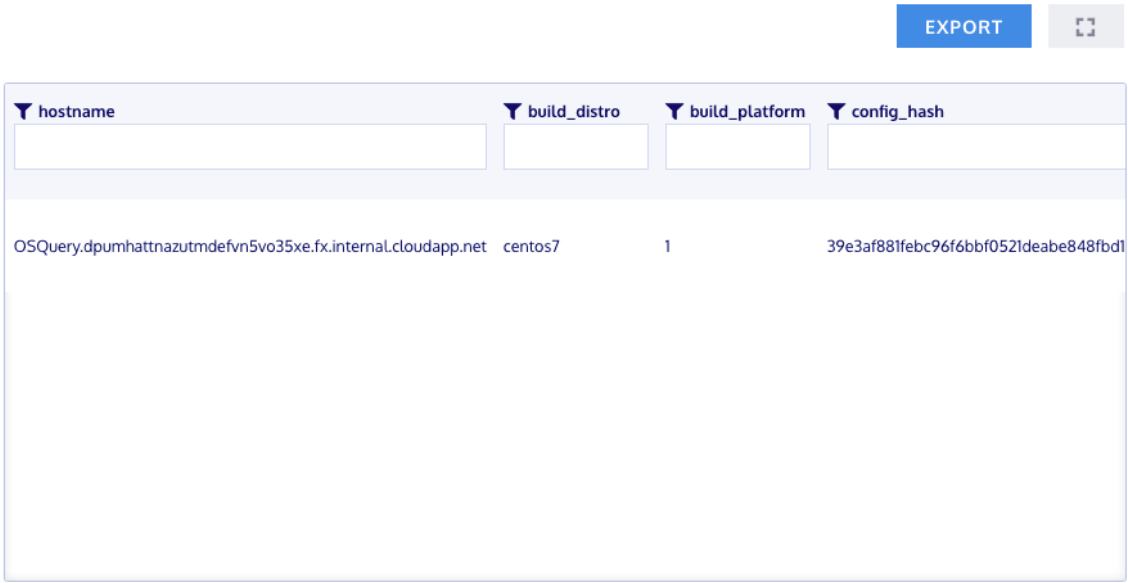
Type the SQL Query



Select the targets/hosts



Click on "Run". You will get the query outputs below:



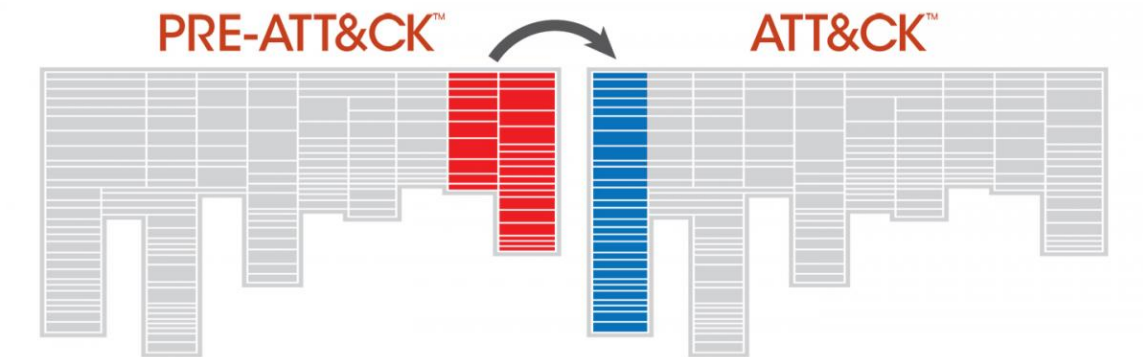
References

1. https://medium.com/@sroberts/osquery-101-getting-started-78e063c4e2f7
2. https://www.digitalocean.com/community/tutorials/how-to-monitor-your-system-security-with-osquery-on-ubuntu-16-04

How to use the MITRE PRE-ATT&CK framework to enhance your reconnaissance assessments

In this module we are going to explore how to enrich reconnaissance assessments using the MITRE Pre-ATT&CK framework.

MITRE ATT&CK Framework



MITRE ATT&CK is a framework developed by the Mitre Corporation. The comprehensive document classifies adversary attacks, in other words, their techniques and tactics after observing millions of real-world attacks against many different organizations. This is why ATT&CK refers to "Adversarial Tactics, Techniques & Common Knowledge".

Nowadays the frameworks provide different matrices: Enterprise, Mobile, and PRE-ATT&CK. Each matrix contains different tactics and each tactic has many techniques.

According to its official website:

> Building on ATT&CK, PRE-ATT&CK provides the ability to prevent an attack before the adversary has a chance to get in. The 15 tactic categories for PRE-ATT&CK were derived from the first two stages (recon and weaponize) of a seven-stage Cyber Attack Lifecycle (first articulated by Lockheed Martin as the Cyber Kill Chain)

The Cyber Kill Chain is a military inspired model to describe the required steps and stages to perform attacks. The Cyber Kill Chain framework is created by Lockheed Martin as part of the Intelligence Driven Defense model for identification and prevention of cyber intrusions activity.

But wait, what is a **tactic** and what is a **technique**?

Tactics, Techniques and procedures (TTPs) are how the attackers are going to achieve their mission. A tactic is the highest level of attack behaviour. The PRE-ATT&CK MITRE framework present the 15 tactics as the following:

1. **Priority Definition Planning**
2. **Priority Definition Direction**
3. **Target Selection**
4. **Technical Information Gathering**
5. **People Information Gathering**
6. **Organizational Information Gathering**
7. **Technical Weakness Identification**
8. **People Weakness Identification**
9. **Organizational Weakness Identification**
10. **Adversary OPSEC**
11. **Establish & Maintain Infrastructure**
12. **Persona Development**
13. **Build Capabilities**
14. **Test Capabilities**
15. **Stage Capabilities**

Techniques are used to execute an attack successfully. PRE-ATT&CK frameworks presents 174 techniques

You can find all the techniques here: https://attack.mitre.org/techniques/pre/

You can find the full matrix (Techniques and tactics) here: https://attack.mitre.org/tactics/pre/

PRE-ATT&CK Matrix

Below are the tactics and techniques representing the MITRE PRE-ATT&CK Matrix.

Last Modified: 2018-04-18 17:59:24.739000
version permalink

Priority Definition Planning	Priority Definition Direction	Target Selection	Technical Information Gathering	People Information Gathering	Organizational Information Gathering	Technical Weakness Identification	People Weakness Identification	Organizational Weakness Identification	Adversary OPSEC	Establish & Maintain Infrastructure	Persona Development	Build Capabilities	Test Capabilities
Assess current holdings, needs, and wants	Assign KITs, KIQs, and/or intelligence requirements	Determine approach/attack vector	Acquire OSINT data sets and information	Acquire OSINT data sets and information	Acquire OSINT data sets and information	Analyze application security posture	Analyze organizational skillsets and deficiencies	Analyze business processes	Acquire and/or use 3rd party infrastructure services	Acquire and/or use 3rd party infrastructure services	Build social network persona	Build and configure delivery systems	Review logs and residual traces
Assess KITs/KIQs benefits	Receive KITs/KIQs and determine requirements	Determine highest level tactical element	Conduct active scanning	Aggregate individual's digital footprint	Conduct social engineering	Analyze architecture and configuration posture	Analyze social and business relationships, interests, and affiliations	Analyze organizational skillsets and deficiencies	Acquire and/or use 3rd party software services	Acquire and/or use 3rd party software services	Choose pre-compromised mobile app developer account credentials or signing keys	Build or acquire exploits	Test ability to evade automated mobile application security analysis performed by app stores
Assess leadership areas of interest	Submit KITs, KIQs, and intelligence requirements	Determine operational element	Conduct passive scanning	Conduct social engineering	Determine 3rd party infrastructure services	Analyze data collected	Assess targeting options	Analyze presence of outsourced capabilities	Acquire or compromise 3rd party signing certificates	Acquire or compromise 3rd party signing certificates	Choose pre-compromised persona and affiliated accounts	C2 protocol development	Test callback functionality
Assign KITs/KIQs into categories	Task requirements	Determine secondary level tactical element	Conduct social engineering	Identify business relationships	Determine centralization of IT management	Analyze hardware/software security defensive capabilities		Assess opportunities created by business deals	Anonymity services	Buy domain name	Develop social network persona digital footprint	Compromise 3rd party or closed-source vulnerability/exploit information	Test malware in various execution environments
Conduct cost/benefit analysis		Determine strategic target	Determine 3rd party infrastructure services	Identify groups/roles	Determine physical locations	Analyze organizational skillsets and deficiencies		Assess security posture of physical locations	Common, high volume protocols and software	Compromise 3rd party infrastructure to support delivery	Friend/Follow/Connect to targets of interest	Create custom payloads	Test malware to evade detection
Create implementation plan			Determine domain and IP address space	Identify job postings and needs/gaps	Dumpster dive	Identify vulnerabilities in third-party software libraries		Assess vulnerability of 3rd party vendors	Compromise 3rd party infrastructure to support delivery	Create backup infrastructure	Obtain Apple iOS enterprise distribution key pair and certificate	Create infected removable media	Test physical access

Now let's explore some techniques:

T1279 Conduct social engineering

Social engineering is the art of hacking humans. In other words, it is a set of techniques (technical and nontechnical) used to get useful and sensitive information from others using psychological manipulation. These are some causes why people and organizations are vulnerable to Social engineering attacks:

- Trust
- Fear
- Greed
- Wanting to help others
- Lack of knowledge

Other causes were discussed and named " **Cialdini's 6 Principles of Influence"**

Cialdini's 6 Principles of Influence:

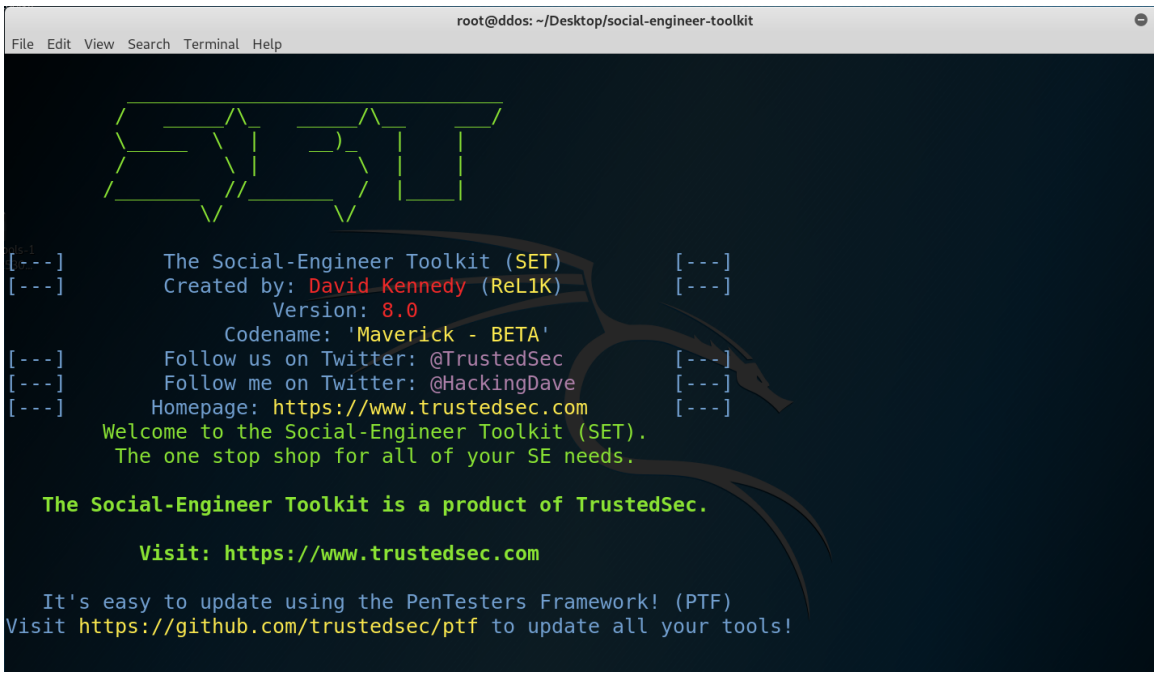
The Cialdini's 6 principles of influence were developed by Dr Robert Cialdini. These principles can be exploited while performing social engineering engagement. The principles are:

1. **Reciprocity:** we pay back what we received from others.
2. **Commitment & Consistency:** We tend to stick with whatever we've already chosen
3. **Social Proof:** We tend to have more trust in things that are popular or endorsed by people that we trust

4. **Liking** We are more likely to comply with requests made by people we like
5. **Authority** : We follow people who look like they know what they're doing
6. **Scarcity:** We are always drawn to things that are exclusive and hard to come by

To perform computer-based social engineering attacks you can use SEToolkit

Social engineering Toolkit is an amazing open source project developed by **Trustedsec** to help penetration testers and ethical hackers perform social engineering attacks. To check the project official GitHub repository you can visit this link: https://github.com/trustedsec/social-engineer-toolkit



T1254 Conduct active scanning

Active reconnaissance involves interaction with the target, for example, calling technical support to gain some sensitive information.Reconnaissance is not only technical. It is also an important weapon of competitive intelligence. Knowing some financial aspects of the target could mean that the attack succeeds. An example of active reconnaissance is network scanning.The aim of network scanning is identifying the live hosts, including the network services of an organization.

To perform network scanning you can use Nmap:

> “Nmap ("Network Mapper") is a free and open source (license) utility for network discovery and security auditing. Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.

T1253 Conduct passive scanning

Passive reconnaissance involves acquiring information about the target without directly interacting with it, for example, searching public information.

T1247 Acquire OSINT data sets and information

By definition:

> "Open-source intelligence (OSINT) is data collected from publicly available sources to be used in an intelligence context". In the intelligence community, the term "open" refers to overt, publicly available sources (as opposed to covert or clandestine sources). It is not related to open-source software or public intelligence.

Open source intelligence is like any methodological process is going thru a defined number of steps.In order to perform an open source intelligence you can follow the following phases:

- **Direction and planning:** in this phase you need to identify the sources,in other words where you can find information
- **Collection:** in this phase you will collect and harvest information from the selected sources
- **Processing and collation:** during this phase you need to process information to get useful insights.
- **Analysis and integration:** in this phase you need to join all the information and analyse them

- **Production, dissemination and feedback:** finally when you finish the analysis you need to present the findings and report them.

One of the available OSINT datasets is **Global Terrorism Database**

During many OSINT missions, you will be dealing with terrorism threats. Thus, it is essential to collect many pieces of information about terrorism online. One of the most used services is the "Global Terrorism Database". The project is managed by the National Consortium for the Study of Terrorism and Responses to Terrorism (START) and it contains information about more than 190,000 terrorist attacks.

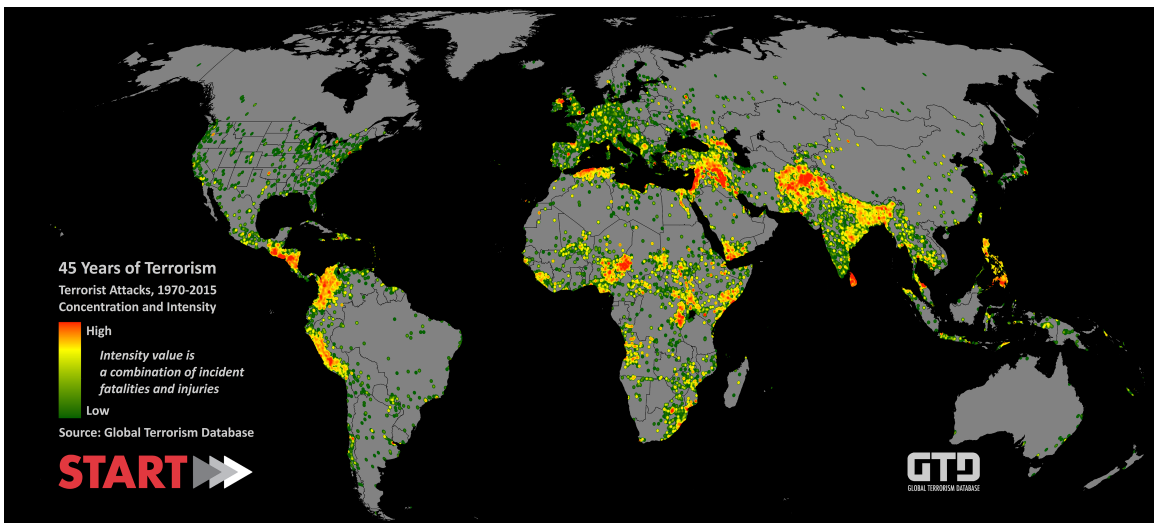


Image source: theconversation.com

T1250 Determine domain and IP address space

To obtain information about the domains, subdomains, IP addresses of the targeted organization you can use https://spyse.com

T1258 Determine Firmware version

Firmware is a set of software that takes control of the device's hardware. You can use a lot of tools and utilities. One of them is binwalk, which is a great tool developed also by Craig Heffner that helps pentesters to analyze the firmware of an IoT device. You can simply grab it from this GitHub link: https://github.com/ReFirmLabs/binwalk/blob/master/INSTALL.md.

```
ddos@ddos ~/Desktop/binwalk
File Edit View Search Terminal Help

Binwalk v2.1.2b
Craig Heffner, http://www.binwalk.org

Usage: binwalk [OPTIONS] [FILE1] [FILE2] [FILE3] ...

Signature Scan Options:
    -B, --signature             Scan target file(s) for common file signatures
    -R, --raw=<str>             Scan target file(s) for the specified sequence of bytes
    -A, --opcodes               Scan target file(s) for common executable opcode signatur
es
    -m, --magic=<file>          Specify a custom magic file to use
    -b, --dumb                  Disable smart signature keywords
    -I, --invalid               Show results marked as invalid
    -x, --exclude=<str>         Exclude results that match <str>
    -y, --include=<str>         Only show results that match <str>

Extraction Options:
    -e, --extract               Automatically extract known file types
    -D, --dd=<type:ext:cmd>     Extract <type> signatures, give the files an extension of
 <ext>, and execute <cmd>
    -M, --matryoshka            Recursively scan extracted files
    -d, --depth=<int>           Limit matryoshka recursion depth (default: 8 levels deep)
    -C, --directory=<str>       Extract files/folders to a custom directory (default: cur
rent working directory)
    -j, --size=<int>            Limit the size of each extracted file
    -n, --count=<int>           Limit the number of extracted files
    -r, --rm                    Delete carved files after extraction
    -z, --carve                 Carve data from files, but don't execute extraction utili
ties
```

T1261 Enumerate externally facing software applications, languages and dependencies

When performing reconnaissance, it is essential to identify the used technologies. For examlpe, to identify the used web technologies you can use: https://www.wappalyzer.com

Image source: https://medium.com/@hari_kishore

T1248 Identify Job postings and needs/gaps

job announcements could be a valuable source of information. Job postings can give an idea about the used systems, technologies and products. To do so, you can check many job boards including:

- Indeed
- Glassdoor

- LinkedIn

T1256 Identify web defensive services

A web application firewall (WAF) is a security solution that filters out bad HTTP traffic between a client and web application. It is a common security control to help you protect your web application security. Most Web application firewalls are helping you to defend against many of the previously discussed web application vulnerabilities (XSS, SQLi and so on). For example to detect WAFs you can use https://github.com/EnableSecurity/wafw00f

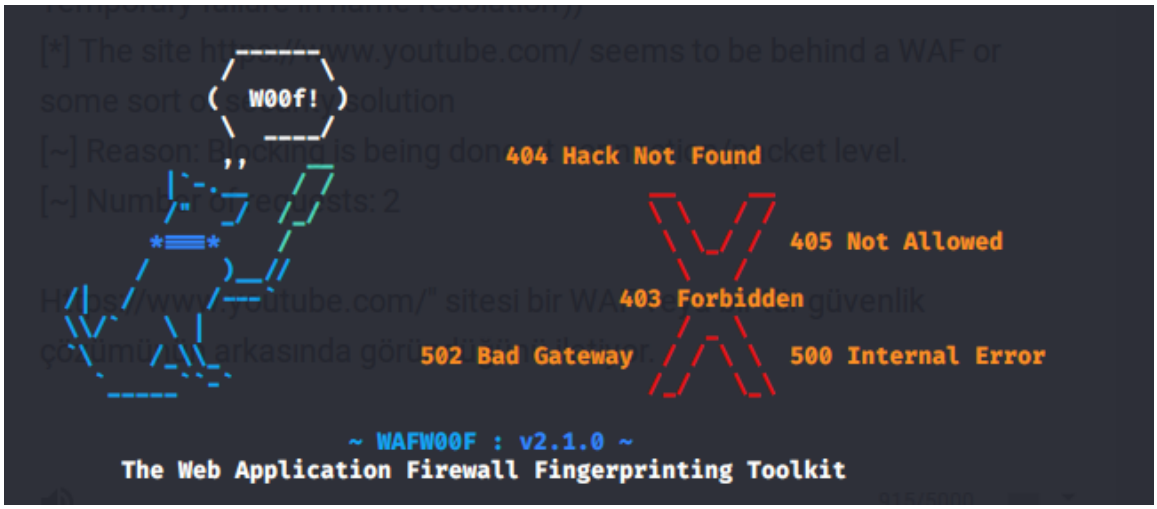
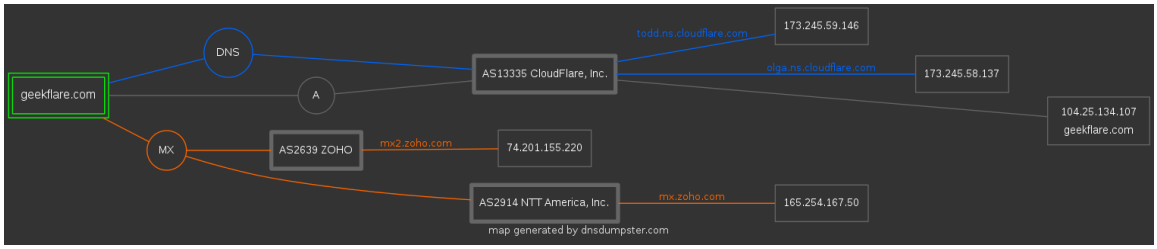


Image source: offensivesec.blogspot.com

T1252 Map network topology

To map network topology you can use many online tools including: https://dnsdumpster.com



T1257 Mine technical blogs/forums

By searching online blogs and technical forums you can collect many useful pieces of information about the targeted organization

T1251 Obtain domain/IP registration information

The Whois database is a publicly accessible database containing the contact details of the owner and contact person of each domain name as well as the data of the name server. It is

usually possible to find out the address, phone number, and e-mail address of the person who owned or at least registered the website. In most cases, this person is the system administrator of the website. You can use this online service: https://whois.net

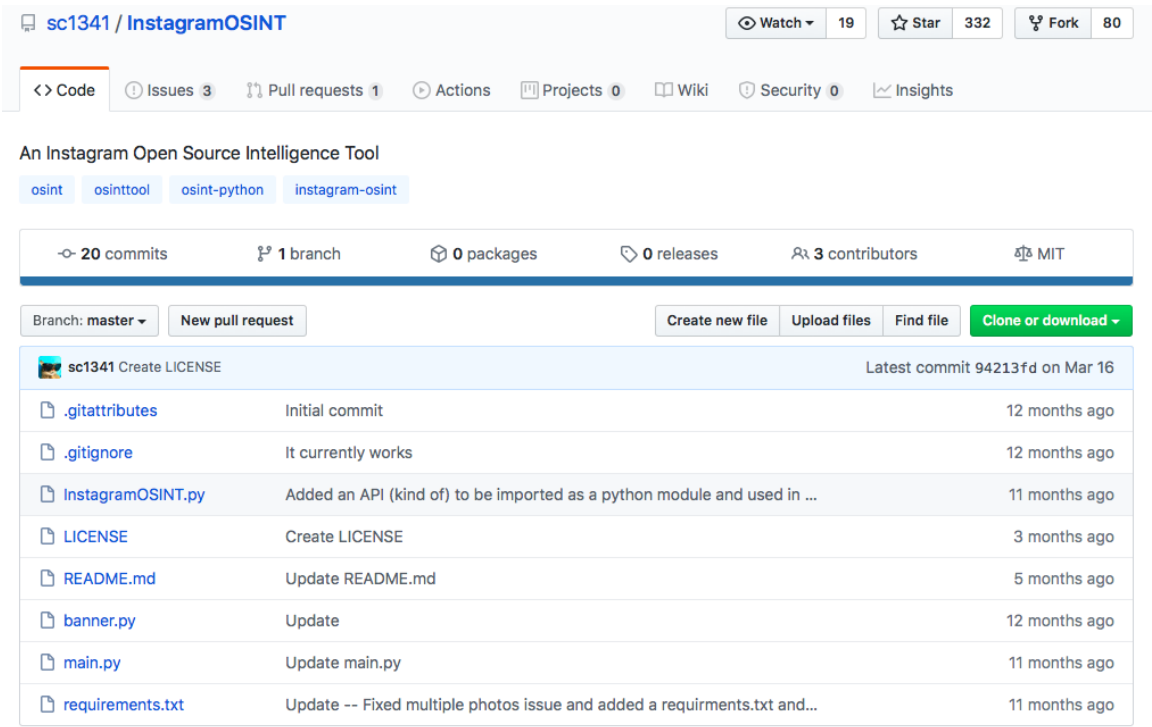
T1271 Identify personnel with an authority/privilege

Generally, it is hard to attack the target directly. Instead, the attackers target employees who have access to the systems, and in particular those with elevated privileges on the target systems. For example, a system administrator would be a great target. To find personnel with authority you can use Linkedin search option.

T1273 Mine social media

When performing open-source intelligence (OSINT), you usually try to find information about people from different publicly available social media platforms including: Facebook, Linkedin, Instagram and so on… To do so, you can use these powerful tools and websites:

- An Instagram Open source Intelligence Tool: https://github.com/sc1341/InstagramOSINT
- Facebook Search tool: https://netbootcamp.org/facebook.html



T1291 Research relevant vulnerabilities/CVEs

This technique consists of finding known vulnerabilities in the targeted systems and applications. Vulnerabilities can be classified using a ranking system, for example, using the **Common Vulnerability Scoring System** (**CVSS**) for the **Common Vulnerabilities and**

Exposures (**CVE**) vulnerabilities. To find vulnerabilities in a service you can use shodan or any vulnerability scanner

Shodan is a search engine that lets the user find specific types of computers (webcams, routers, servers, etc.) connected to the internet using a variety of filters. Some have also described it as a search engine of service banners, which are metadata that the server sends back to the client. This can be information about the server software, what options the service supports, a welcome message or anything else that the client can find out before interacting with the server.

⚠ Vulnerabilities

Note: the device may not be impacted by all of these issues. The vulnerabilities are implied based on the software and version.

CVE-2011-5000	The ssh_gssapi_parse_ename function in gss-serv.c in OpenSSH 5.8 and earlier, when gssapi-with-mic authentication is enabled, allows remote authenticated users to cause a denial of service (memory consumption) via a large value in a certain length field. NOTE: there may be limited scenarios in which this issue is relevant.
CVE-2010-4478	OpenSSH 5.6 and earlier, when J-PAKE is enabled, does not properly validate the public parameters in the J-PAKE protocol, which allows remote attackers to bypass the need for knowledge of the shared secret, and successfully authenticate, by sending crafted values in each round of the protocol, a related issue to CVE-2010-4252.
CVE-2014-1692	The hash_buffer function in schnorr.c in OpenSSH through 6.4, when Makefile.inc is modified to enable the J-PAKE protocol, does not initialize certain data structures, which might allow remote attackers to cause a denial of service (memory corruption) or have unspecified other impact via vectors that trigger an error condition.
CVE-2010-5107	The default configuration of OpenSSH through 6.1 enforces a fixed time limit between establishing a TCP connection and completing a login, which makes it easier for remote attackers to cause a denial of service (connection-slot exhaustion) by periodically making many new TCP connections.
CVE-2017-15906	The process_open function in sftp-server.c in OpenSSH before 7.6 does not properly prevent write operations in readonly mode, which allows attackers to create zero-length files.
CVE-2016-10708	sshd in OpenSSH before 7.4 allows remote attackers to cause a denial of service (NULL pointer dereference and daemon crash) via an out-of-sequence NEWKEYS message, as demonstrated by Honggfuzz, related to kex.c and packet.c.
CVE-2016-0777	The resend_bytes function in roaming_common.c in the client in OpenSSH 5.x, 6.x, and 7.x before 7.1p2 allows remote servers to obtain sensitive information from process memory by requesting transmission of an entire buffer, as demonstrated by reading a private key.
CVE-2011-4327	ssh-keysign.c in ssh-keysign in OpenSSH before 5.8p2 on certain platforms executes ssh-rand-helper with unintended open file descriptors, which allows local users to obtain

Summary

In this module we explored the MITRE PRE-ATT&CK framework and we discovered some techniques used when performing reconnaissance against an organization

How to Perform Open Source Intelligence (OSINT) with SpiderFoot

In this module we are going to explore a powerful OSINT tool called "SpiderFoot". OSINT or "Open source intelligence" is collecting publicly available information about a specific target.

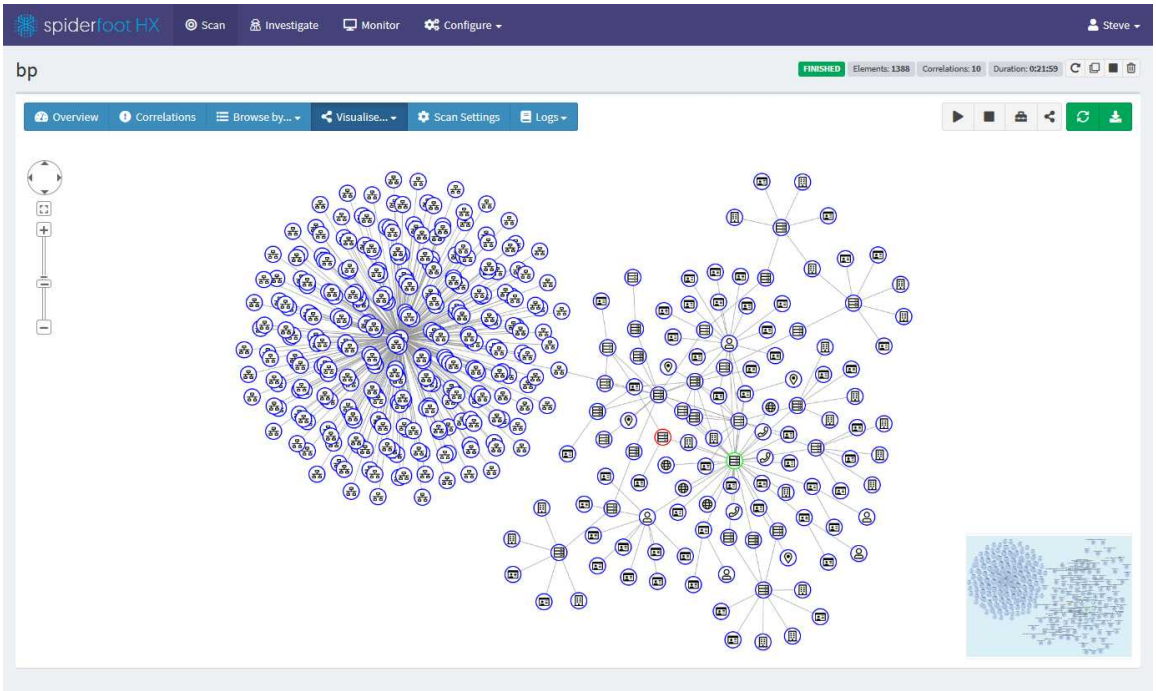


Image source

Before discovering the tool, let's explore some important terminologies

Intelligence

The fuel of intelligence gathering is to get publicly available information from different sources. Intelligence gathering is not important in information security and penetration testing, but it is vital for national security, and as many concepts are inspired by the military strategies, in the cyber security field intelligence gathering is also inspired by the battlefields.

Image source

According to International Trade Commission estimates, current annual losses to US industries due to corporate espionage to be over $70 billion.

Intelligence gathering not only helps improve the security position of the organization, but it gives managers an eagle eye on the competition, and it results in better business decisions. Basically every intelligence gathering operation basically is done following a structured methodology.

There are many intelligence gathering categories: human intelligence, signal intelligence, open source intelligence, imagery intelligence, and geospatial intelligence.

Human intelligence (HUMINT)

Human intelligence (HUMINT) is the process of collecting information about human targets, with or without interaction with them, using many techniques such as taking photographs and video recording. There are three models of human intelligence:

- **Directed Gathering** : This is a specific targeting operation. Usually, all the resources are meant to gather information about a unique target
- **Active Intelligence Gathering** : This process is more specific and requires less investment, and it targets a specific environment.

- **Passive Intelligence Gathering** : This is the foundation of human intelligence. The information is collected in opportunistic ways such as through walk-ins or referrals. So there is no specific target, except collecting information and trying to find something.

Image source

Signal intelligence

Signal intelligence (**SIGINT**) is the operation of gathering information by intercepting electronic signals and communications. It can be divided into two subcategories: **communications intelligence** (**COMINT**) and **electronic intelligence** (**ELINT**).

Open source intelligence

Public intelligence is the process of gathering all possible information about the target, using publicly available sources, and not only searching for it but also archiving it. The term is generally used by government agencies for national security operations. A penetration tester should also adopt such a state of mind and acquire the required skills to gather and classify information. In the era of huge amounts of data, the ability to extract useful information from it is a must.

Open source intelligence (**OSINT**), as its name suggests, involves finding information about a defined target using available sources online. It can be done using many techniques:

Conducting search queries in many search engines Gaining information from social media networks Searching in _deep web _directories and the hidden wiki Using forum and discussion boards

The OSINT process

Open source intelligence is like any methodological process is going thru a defined number of steps.In order to perform an open source intelligence you can follow the following phases:

- **Direction and planning:** in this phase you need to identify the sources,in other words where you can find information
- **Collection:** in this phase you will collect and harvest information from the selected sources
- **Processing and collation:** during this phase you need to process information to get useful insights.
- **Analysis and integration:** in this phase you need to join all the information and analyse them
- **Production, dissemination and feedback:** finally when you finish the analysis you need to present the findings and report them.

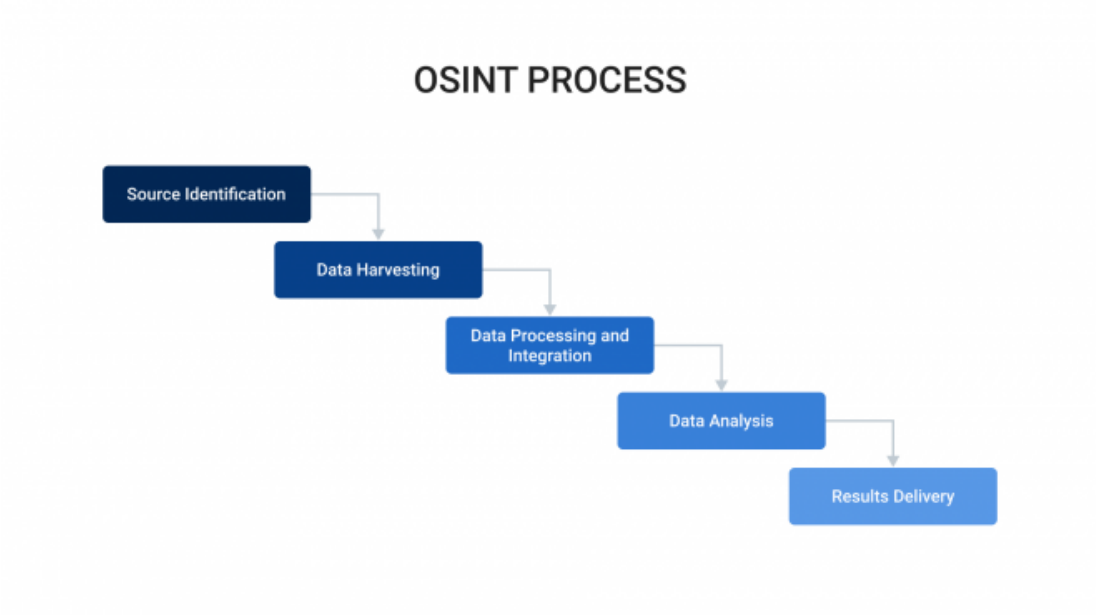


Image source

There are many helpful tools that you can use to perform OSINT, you can find some of them in this post:

How to Deploy SpiderFoot

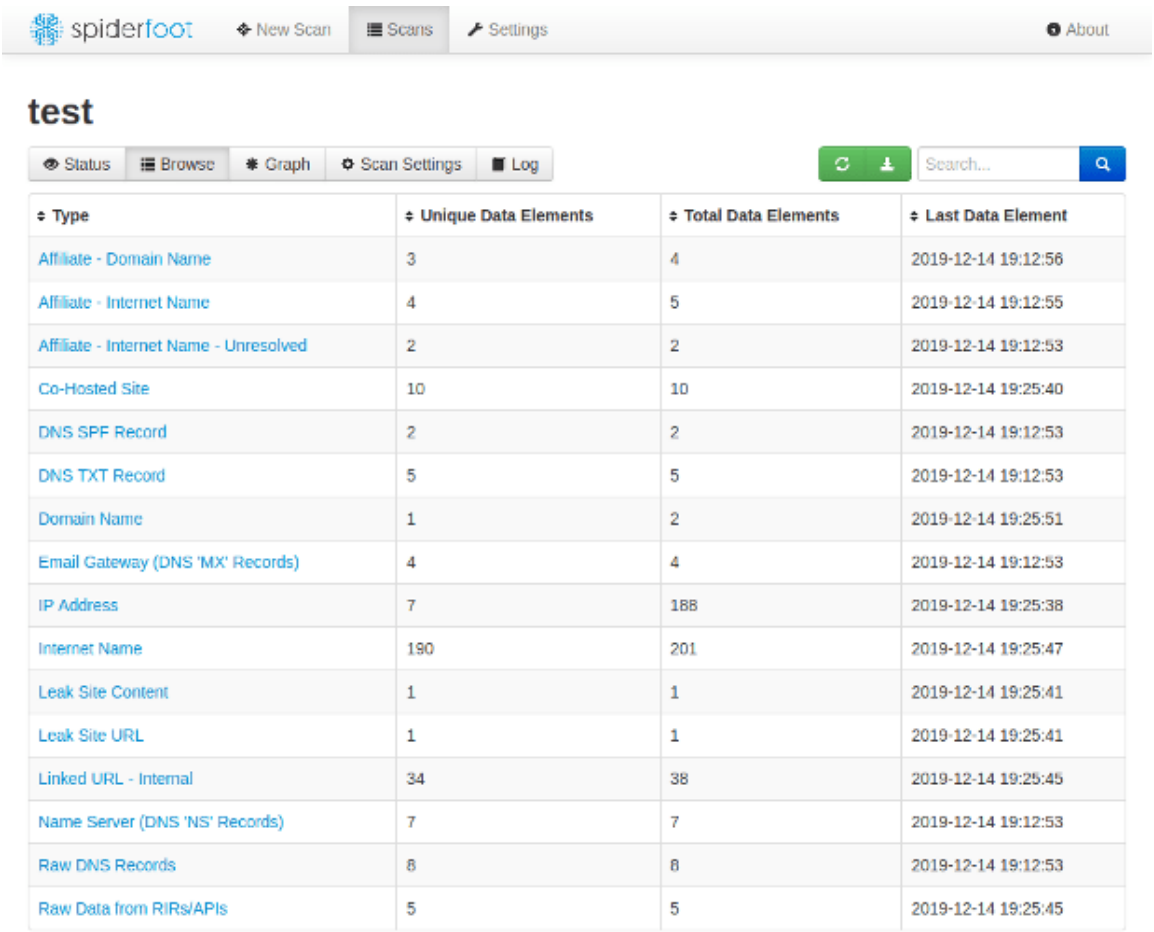
According to its official github repository:

spiderfoot

SpiderFoot __ __ is an open source intelligence (OSINT) __ __ automation __ __ tool. It integrates with just about every data source available and utilises a range of methods for __ __ data

analysis__, making that data easy to navigate.

SpiderFoot has an __ __ embedded __ __ web-server for providing a clean and intuitive __ __ web-based __ __ interface __ __ but can also be used completely via the command-line. It's written in __ __ Python __ __ 3 and GPL-licensed.



Spiderfoot is able to collect information about:

- IP address
- Domain/sub-domain name
- Hostname
- Network subnet (CIDR)
- ASN
- E-mail address
- Phone number
- Username
- Person's name

Now let's explore how to install Spiderfoot.

Install python3-pip:

```
sudo apt-get install python3-pip
```

Clone the project from its Github repository using **git clone** :

```
git clone https://github.com/smicallef/spiderfoot.git
```

Enter the project folder:

```
cd spiderfoot
```

Install the required libraries:

```
sudo pip3 install -r requirements.txt
```

Finally run the project using:

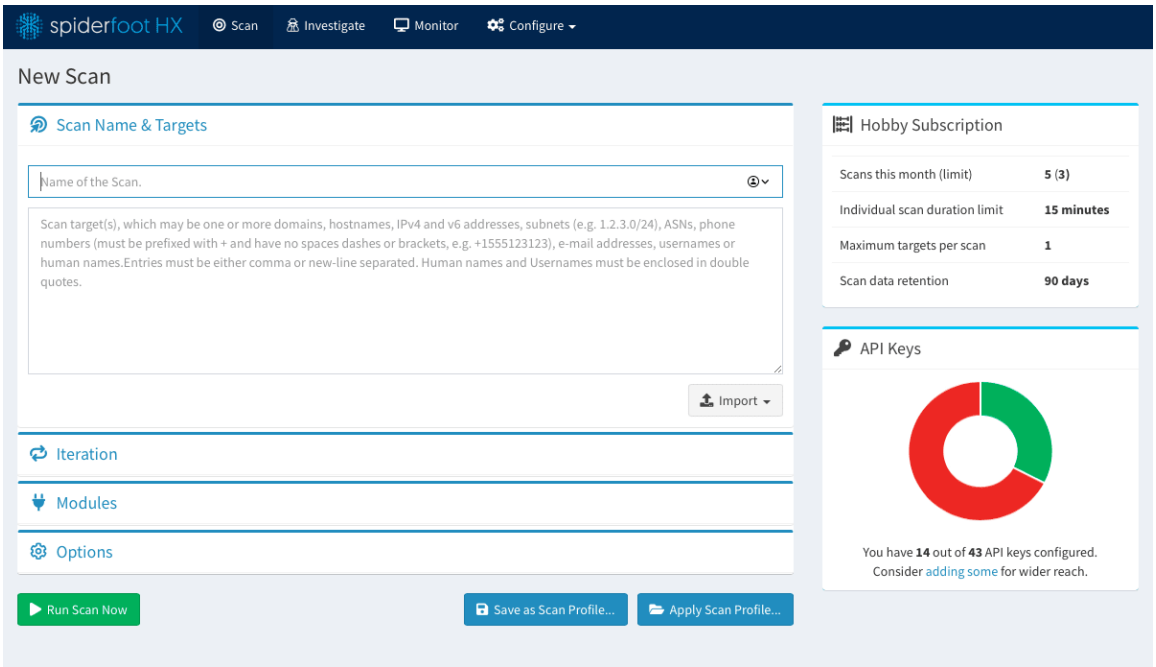
```
sudo python3 sf.py -l 127.0.0.1:5001
```

Voila! Now you can use it freely to perform your OSINT operation.

There is another option which is using a ready-to-go Spiderfoot instance. To do it check this link: https://www.spiderfoot.net/hx/

To start a new scan, click on " **+ Create a new scan**"

Enter your target and click on " **Run scan now**"



As you can notice from the screenshot there are some APIs that need to be added n order to use some modules.

A module is a specific entity that perform a specific task. Spiderfoot comes with a long list of modules including:

- **abuse.ch:** Checks if a host/domain, IP or netblock is malicious according to abuse.ch.
- **Accounts:** Looks for possible associated accounts on nearly 200 websites like Ebay, Slashdot, reddit, etc.
- **AlienVault OTX:** Obtains information from AlienVault Open Threat Exchange (OTX)

The full list of modules can be found here: https://github.com/smicallef/spiderfoot

The tool gives you the ability to investigate data too:

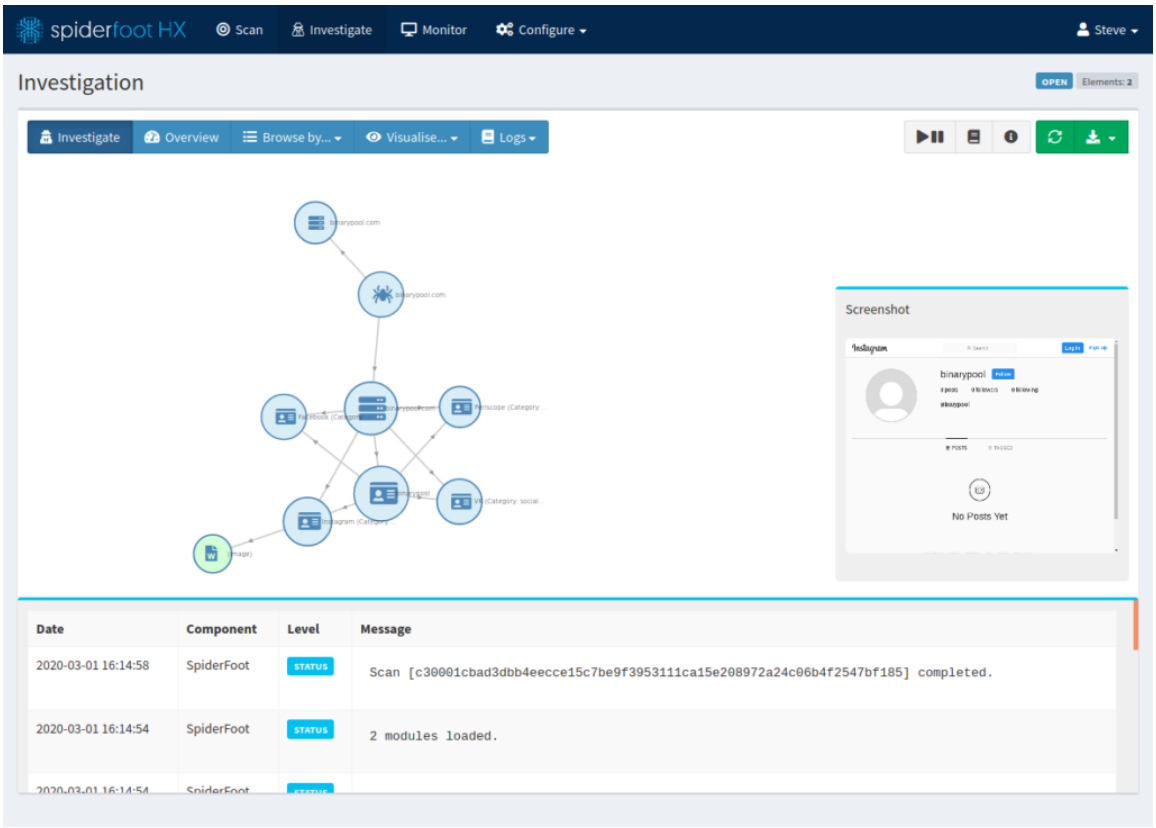


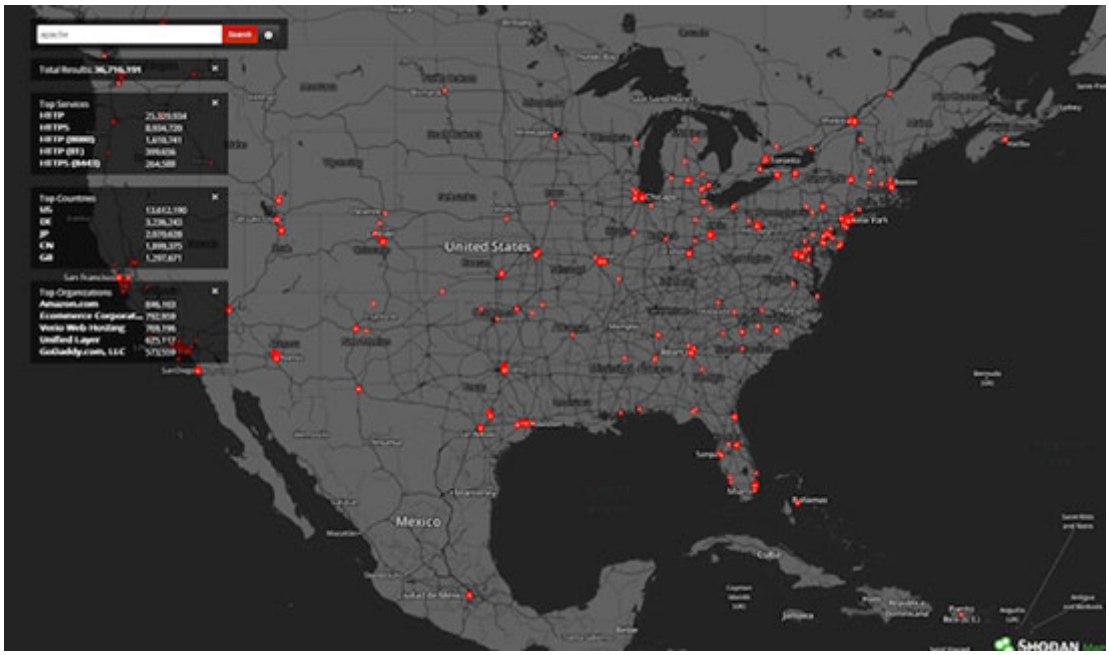
Image source

Summary

In this module, we explored Open source intelligence and how to perform it using a powerful tool called "SpiderFoot"

How to perform OSINT with Shodan

In some of my previous articles we had the opportunity to explore different techniques to perform intelligence gathering including Human intelligence,signal intelligence, Geospatial intelligence and Open source intelligence. In this article we will dive deep into a powerful open source intelligence online tool called Shodan.



What is Open source intelligence?

Wikipedia defines OSINT as follows:

> *"Open-source intelligence is data collected from publicly available sources to be used in an intelligence context. In the intelligence community, the term "open" refers to overt, publicly available sources. It is not related to open-source software or collective intelligence"*

Open source intelligence is like any methodological process is going thru a defined number of steps.In order to perform an open source intelligence you can follow the following phases:

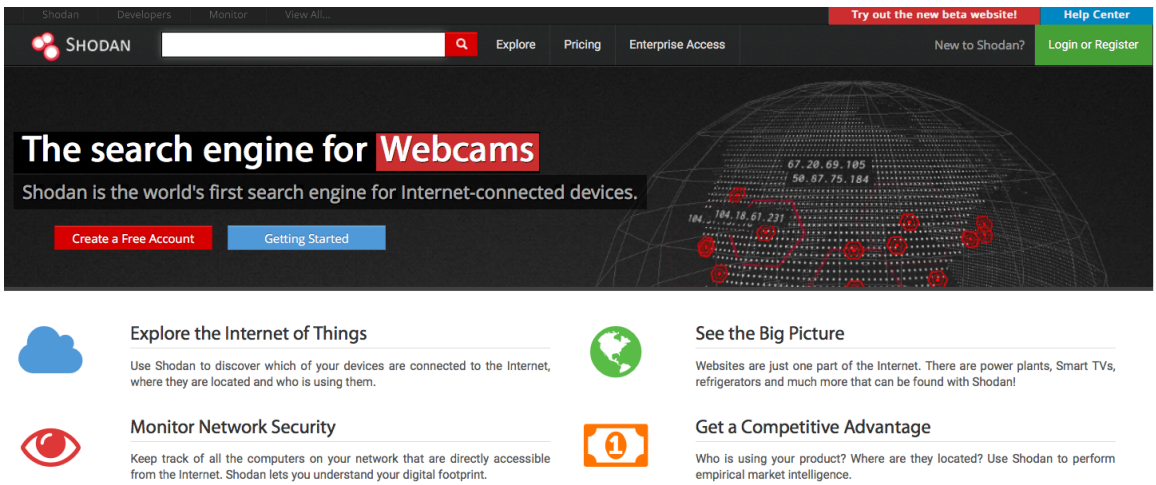
1. Direction and planning: in this phase you need to identify the sources,in other words where you can find information
2. Collection: in this phase you will collect and harvest information from the selected sources
3. Processing and collation: during this phase you need to process information to get useful insights.

4. Analysis and integration: in this phase you need to join all the information and analyse them
5. Production, dissemination and feedback: finally when you finish the analysis you need to present the findings and report them.

What is Shodan?

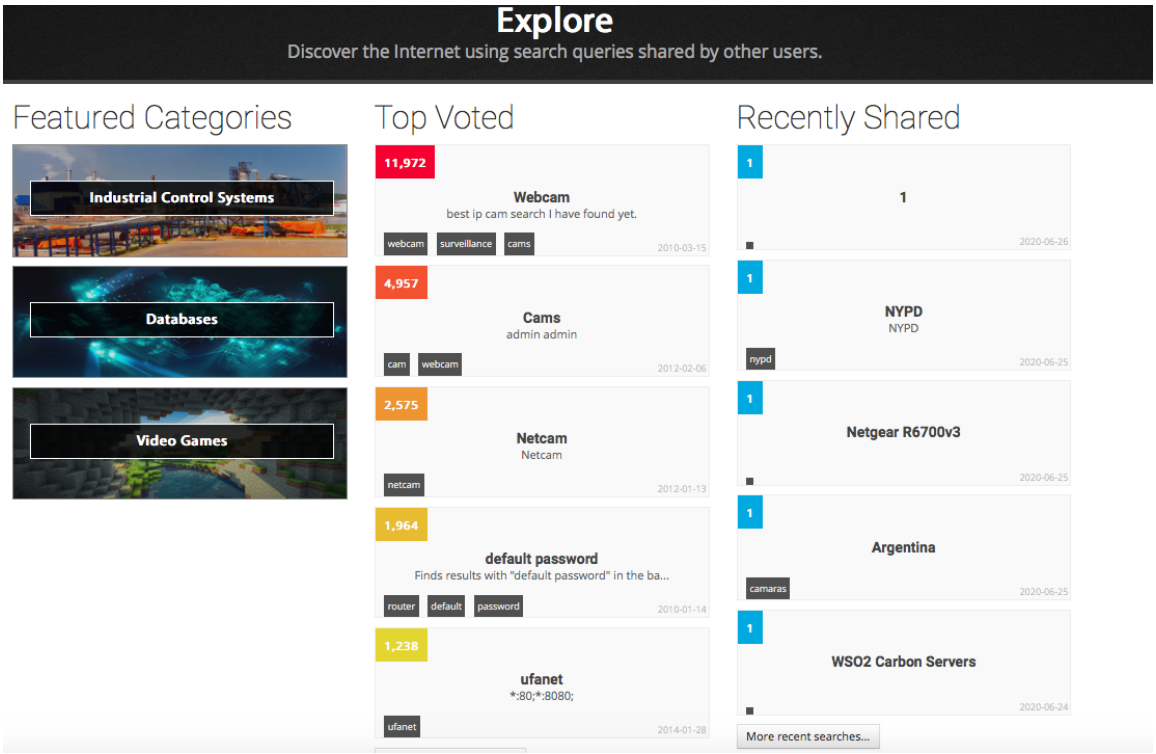
Shodan is a search engine that lets the user find specific types of computers (webcams, routers, servers, etc.) connected to the internet using a variety of filters. Some have also described it as a search engine of service banners, which are metadata that the server sends back to the client. This can be information about the server software, what options the service supports, a welcome message or anything else that the client can find out before interacting with the server.

You can use it by visiting the official website: www.shodan.io



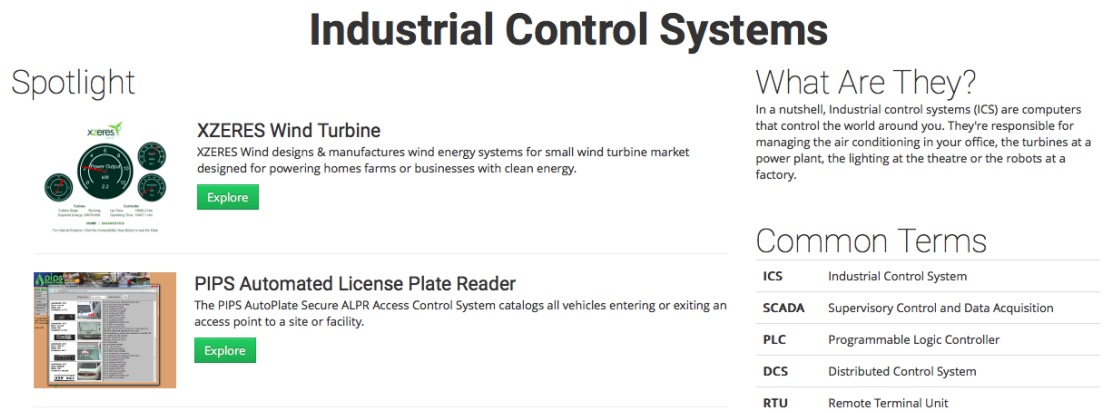
As a start, Shodan gives you the ability to start exploring some pre-selected search queries. Some of the findings are:

- Webcams
- Industrial control systems
- Databases
- Passwords and so on



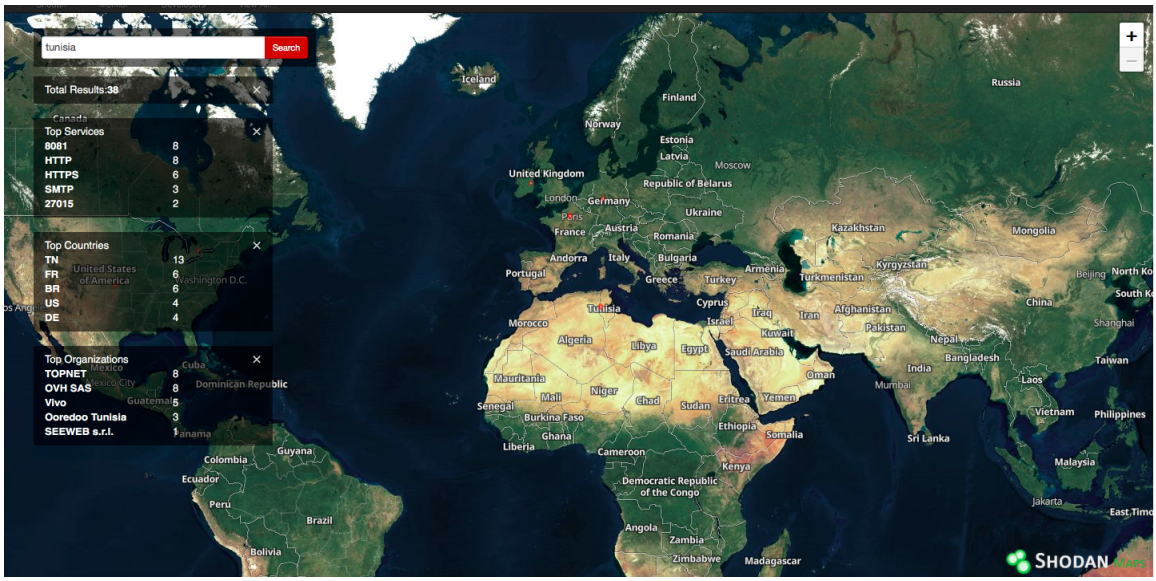
For example, in the Industrial control systems section, you can search for

- XZERES Wind Turbines
- PIPS Automated License Plate Readers



It supports many ICS protocols too.

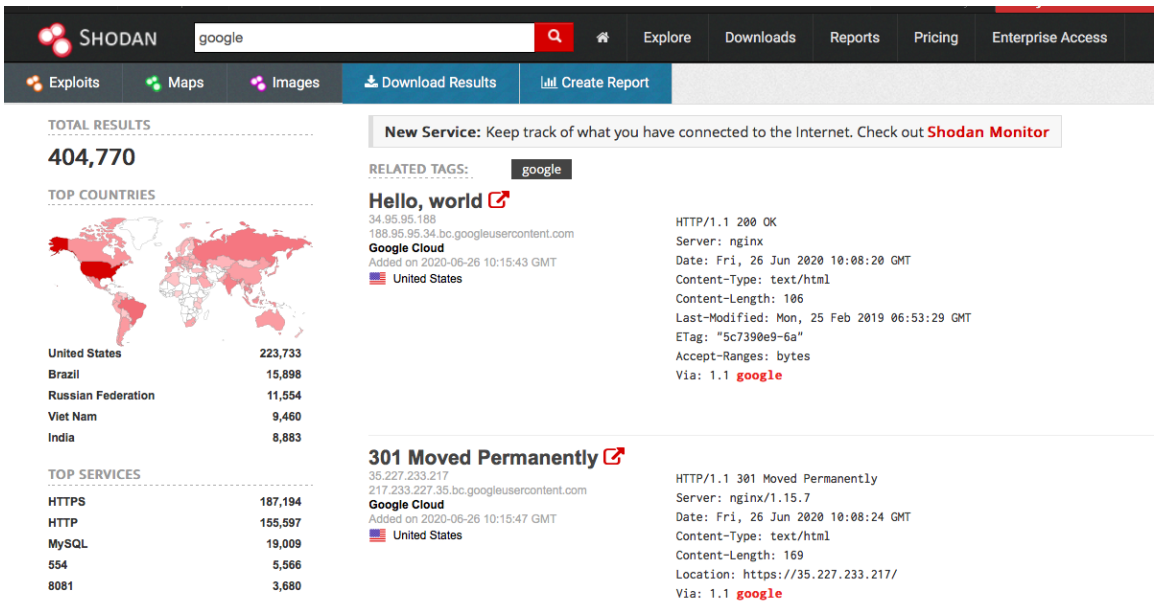
Furthermore, you can use shodan map for more geo-centric searches



Now let's explore how to perform some shodan queries.

To perform search, you will simply use the search bar in the main page

To simpliest search form is typing the "term" you are looking for, like a website name, service or something and shodan will give pages of results that you can filter later



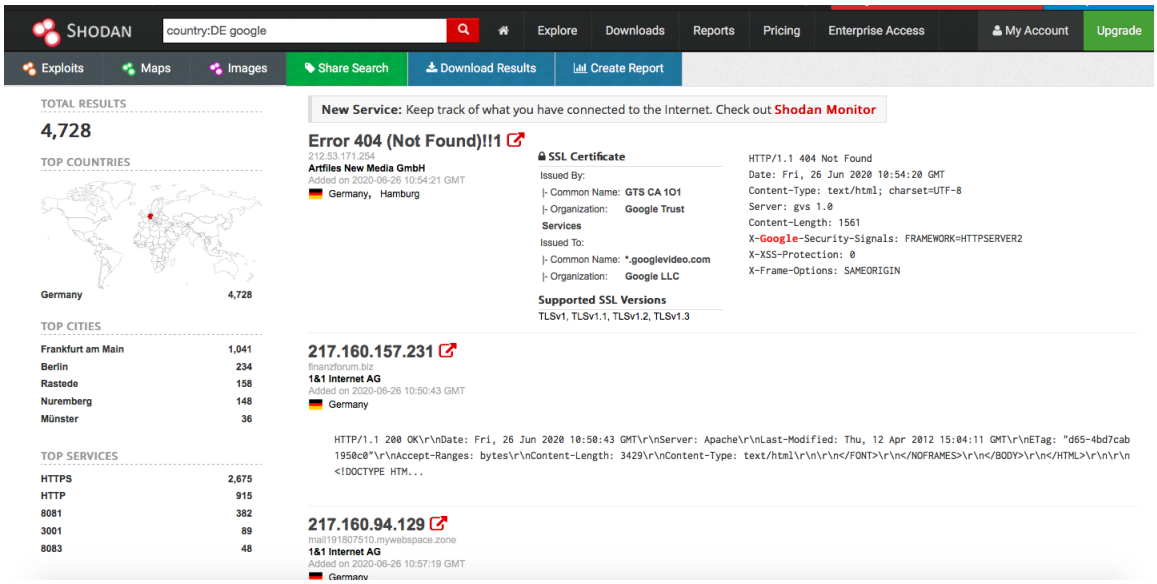
Queries can be more specific. Shodan provides a list of advanced queries that you can use in order to get more accurate information. Some of them are the following:

To select a specific country type:

```
country: <Country Symbol>
```

For example, Germany code is: DE. So the query will be:

```
country:DE
```



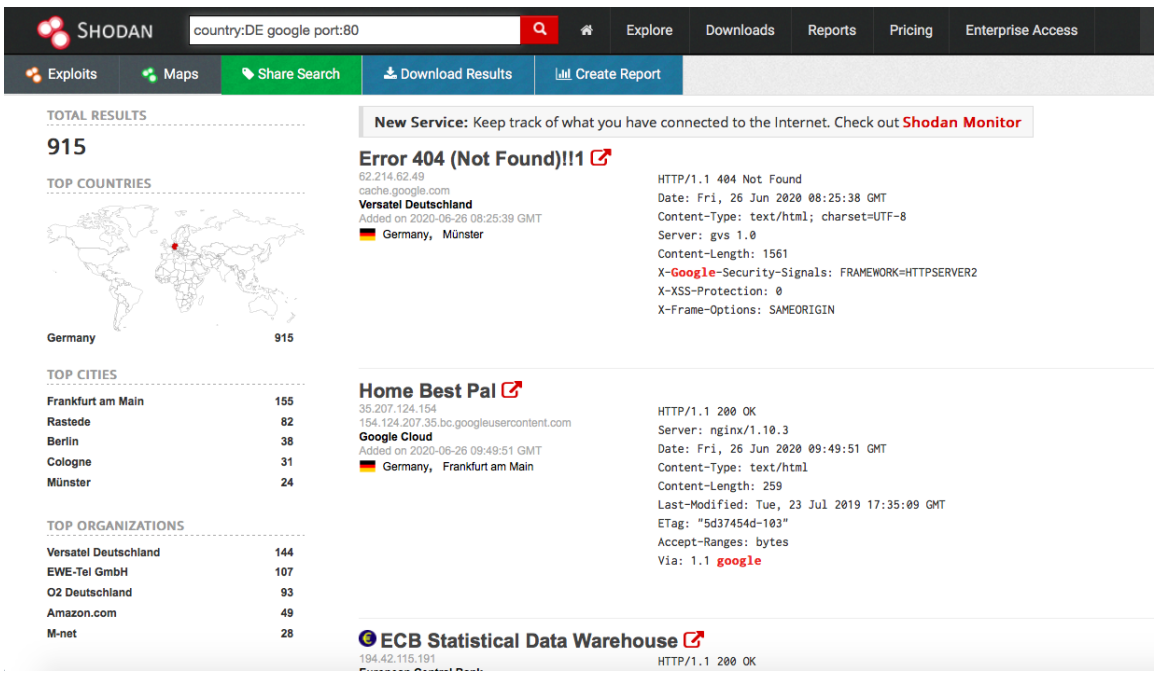
County codes can be found here: https://github.com/postmodern/shodan-ruby/blob/master/lib/shodan/countries.rb

To select specific ports type:

```
port: <Ports_HERE>
```

For example:

```
port:80
```



To search for a specifit operating system(OS) type:

```
os: <OS_HERE>
```

Using MITRE ATT&CK to defend against Advanced Persistent Threats

Nowadays, new techniques are invented on a daily basis to bypass security layers and avoid detection. Thus it is time to figure out new techniques too and defend against cyber threats.

Image Courtesy

Before diving into how to use MITRE ATT&CK framework to defend against advanced persistent threats and protect critical assets, let's explore some important terminologies

Threats

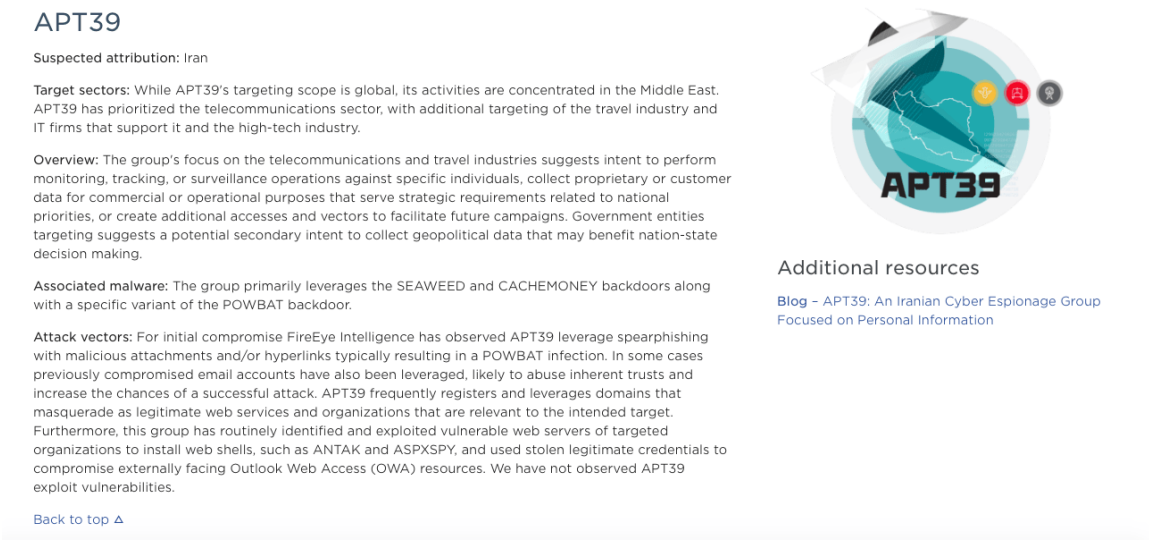
By definition, a **threat** is a potential danger for the enterprise assets that could harm these systems. In many cases, there is confusion between the three terms Threat, Vulnerability and Risk; the first term, as I explained before, is a potential danger while a Vulnerability is a known weakness or a gap in an asset. A risk is a result of a threat exploiting a vulnerability. In other words, you can see it as an intersection between the two previous terms. The method used to attack an asset is called a **Threat Vector**.

Advanced Persistent Threats

Wikipedia defines an "Advanced Persistence Threat" as follows:

> "An advanced persistent threat is a stealthy computer network threat actor, typically a nation-state or state-sponsored group, which gains unauthorized access to a computer network and remains undetected for an extended period"

To discover some of the well-known APT groups you can check this great resource from FireEye: Advanced Persistent Threat Groups



APT39

Suspected attribution: Iran

Target sectors: While APT39's targeting scope is global, its activities are concentrated in the Middle East. APT39 has prioritized the telecommunications sector, with additional targeting of the travel industry and IT firms that support it and the high-tech industry.

Overview: The group's focus on the telecommunications and travel industries suggests intent to perform monitoring, tracking, or surveillance operations against specific individuals, collect proprietary or customer data for commercial or operational purposes that serve strategic requirements related to national priorities, or create additional accesses and vectors to facilitate future campaigns. Government entities targeting suggests a potential secondary intent to collect geopolitical data that may benefit nation-state decision making.

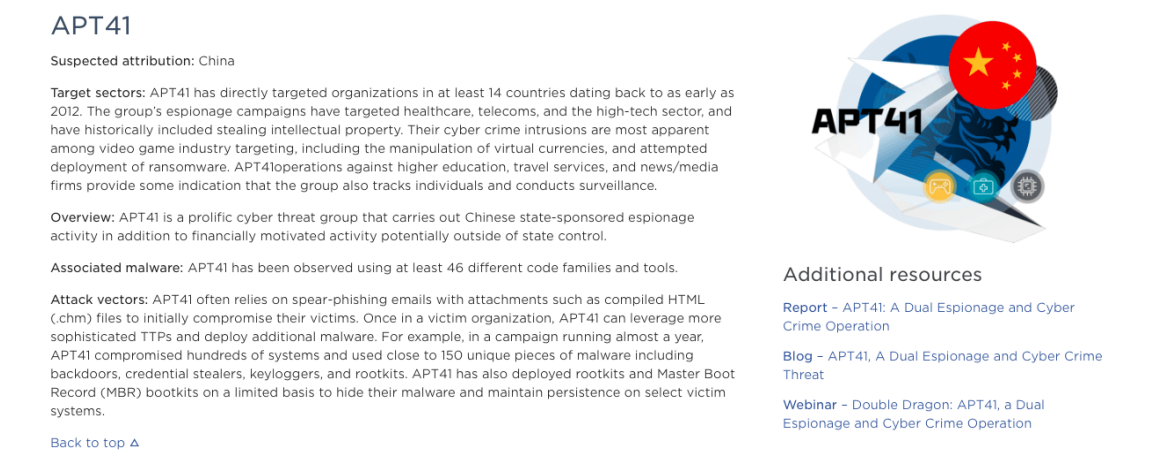
Associated malware: The group primarily leverages the SEAWEED and CACHEMONEY backdoors along with a specific variant of the POWBAT backdoor.

Attack vectors: For initial compromise FireEye Intelligence has observed APT39 leverage spearphishing with malicious attachments and/or hyperlinks typically resulting in a POWBAT infection. In some cases previously compromised email accounts have also been leveraged, likely to abuse inherent trusts and increase the chances of a successful attack. APT39 frequently registers and leverages domains that masquerade as legitimate web services and organizations that are relevant to the intended target. Furthermore, this group has routinely identified and exploited vulnerable web servers of targeted organizations to install web shells, such as ANTAK and ASPXSPY, and used stolen legitimate credentials to compromise externally facing Outlook Web Access (OWA) resources. We have not observed APT39 exploit vulnerabilities.

Back to top ▵

Additional resources

Blog – APT39: An Iranian Cyber Espionage Group Focused on Personal Information



APT41

Suspected attribution: China

Target sectors: APT41 has directly targeted organizations in at least 14 countries dating back to as early as 2012. The group's espionage campaigns have targeted healthcare, telecoms, and the high-tech sector, and have historically included stealing intellectual property. Their cyber crime intrusions are most apparent among video game industry targeting, including the manipulation of virtual currencies, and attempted deployment of ransomware. APT41operations against higher education, travel services, and news/media firms provide some indication that the group also tracks individuals and conducts surveillance.

Overview: APT41 is a prolific cyber threat group that carries out Chinese state-sponsored espionage activity in addition to financially motivated activity potentially outside of state control.

Associated malware: APT41 has been observed using at least 46 different code families and tools.

Attack vectors: APT41 often relies on spear-phishing emails with attachments such as compiled HTML (.chm) files to initially compromise their victims. Once in a victim organization, APT41 can leverage more sophisticated TTPs and deploy additional malware. For example, in a campaign running almost a year, APT41 compromised hundreds of systems and used close to 150 unique pieces of malware including backdoors, credential stealers, keyloggers, and rootkits. APT41 has also deployed rootkits and Master Boot Record (MBR) bootkits on a limited basis to hide their malware and maintain persistence on select victim systems.

Back to top ▵

Additional resources

Report – APT41: A Dual Espionage and Cyber Crime Operation

Blog – APT41, A Dual Espionage and Cyber Crime Threat

Webinar – Double Dragon: APT41, a Dual Espionage and Cyber Crime Operation

The Cyber Kill Chain

The Cyber Kill Chain is a military inspired model to describe the required steps and stages to perform attacks. The Cyber Kill Chain framework is created by Lockheed Martin as part of the Intelligence Driven Defense model for identification and prevention of cyber intrusions activity. While a kill chain in military refers to: Find, Fix, Track, Target, Engage and Assess, cyber kill chain refers to: reconnaissance, Initial attack, Command and control, Discover and spread and finally Extraction and exfiltration. Knowing this framework is essential to have a clearer understanding about how major attacks occur.

Image Courtesy

Threat intelligence is an important operation in cyber-security and especially in security operations and incident response. Because as Sun Tzu said:

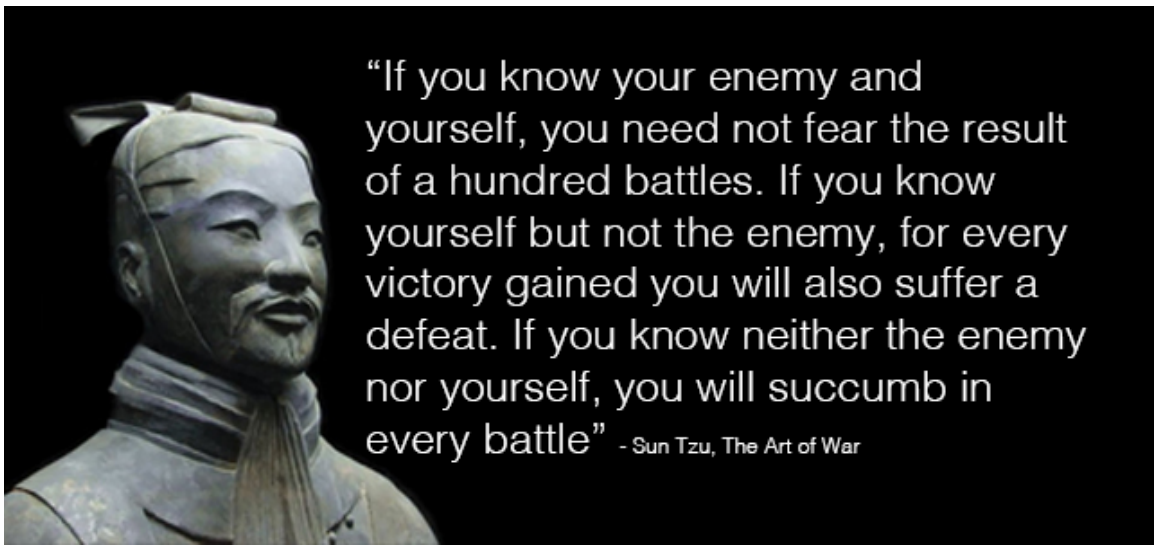


Image Courtesy

Security operation analysts should be proactive when it comes to gathering information and intelligence about the external threats and adversaries to achieve faster detection.

MITRE ATT&CK Framework

ATT&CK™
Adversarial Tactics, Techniques & Common Knowledge

MITRE ATT&CK is a framework developed by the Mitre Corporation. The comprehensive document classifies adversary attacks, in other words, their techniques and tactics after observing millions of real-world attacks against many different organizations. This is why ATT&CK refers to "Adversarial Tactics, Techniques & Common Knowledge".

Nowadays the frameworks provide different matrices: Enterprise, Mobile, and PRE-ATT&CK. Each matrix contains different tactics and each tactic has many techniques.

But wait, what is a **tactic** and what is a **technique**?

To understand tactics and techniques we need to understand the pyramid of pain first. The pyramid of pain shows the relationship between the types of indicators found when dealing

with adversaries. By indicators, I mean Hash values, IP addresses, Domain names, Network/host artefacts, tools and Tactics, techniques and procedures (TTPs).

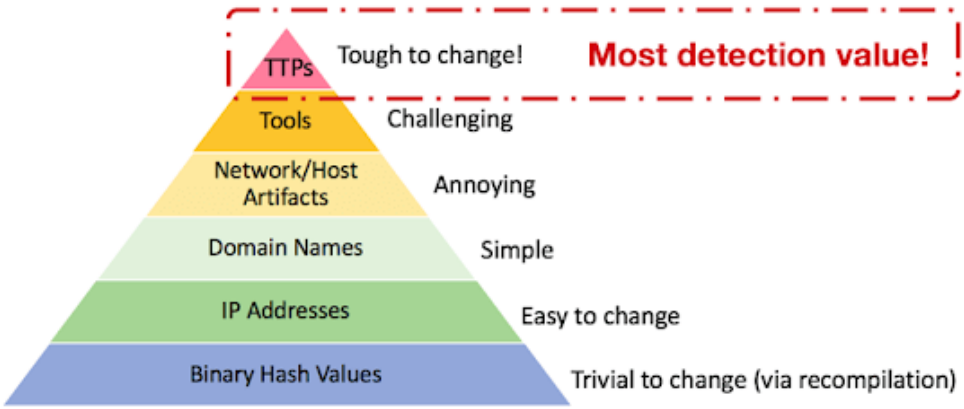


Image Courtesy

Tactics, Techniques and procedures (TTPs) are how the attackers are going to achieve their mission. A tactic is the highest level of attack behaviour. MITRE framework present the tactics as the following:

1. **Initial Access**
2. **Execution**
3. **Persistence**
4. **Privilege Escalation**
5. **Defense Evasion**
6. **Credential Access**
7. **Discovery**
8. **Lateral Movement**
9. **Collection**
10. **Exfiltration**
11. **Command and Control**

Techniques are used to execute an attack successfully. For example, this is information about the "*AppCertDLLs*" technique

Home > Techniques > Enterprise > AppCert DLLs

AppCert DLLs

Dynamic-link libraries (DLLs) that are specified in the AppCertDLLs Registry key under `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager` are loaded into every process that calls the ubiquitously used application programming interface (API) functions CreateProcess, CreateProcessAsUser, CreateProcessWithLoginW, CreateProcessWithTokenW, or WinExec. [1]

Similar to Process Injection, this value can be abused to obtain persistence and privilege escalation by causing a malicious DLL to be loaded and run in the context of separate processes on the computer.

ID: T1182

Tactic: Persistence, Privilege Escalation

Platform: Windows

Permissions Required: Administrator, SYSTEM

Effective Permissions: Administrator, SYSTEM

Data Sources: Loaded DLLs, Process monitoring, Windows Registry

Version: 1.0

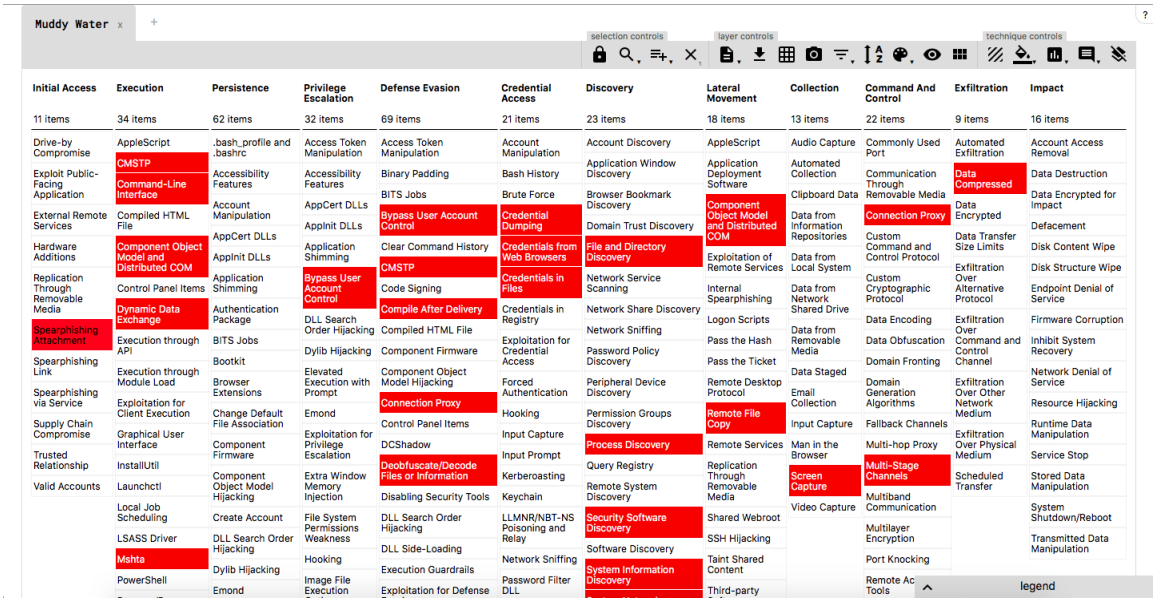
Created: 16 January 2018

Last Modified: 16 July 2019

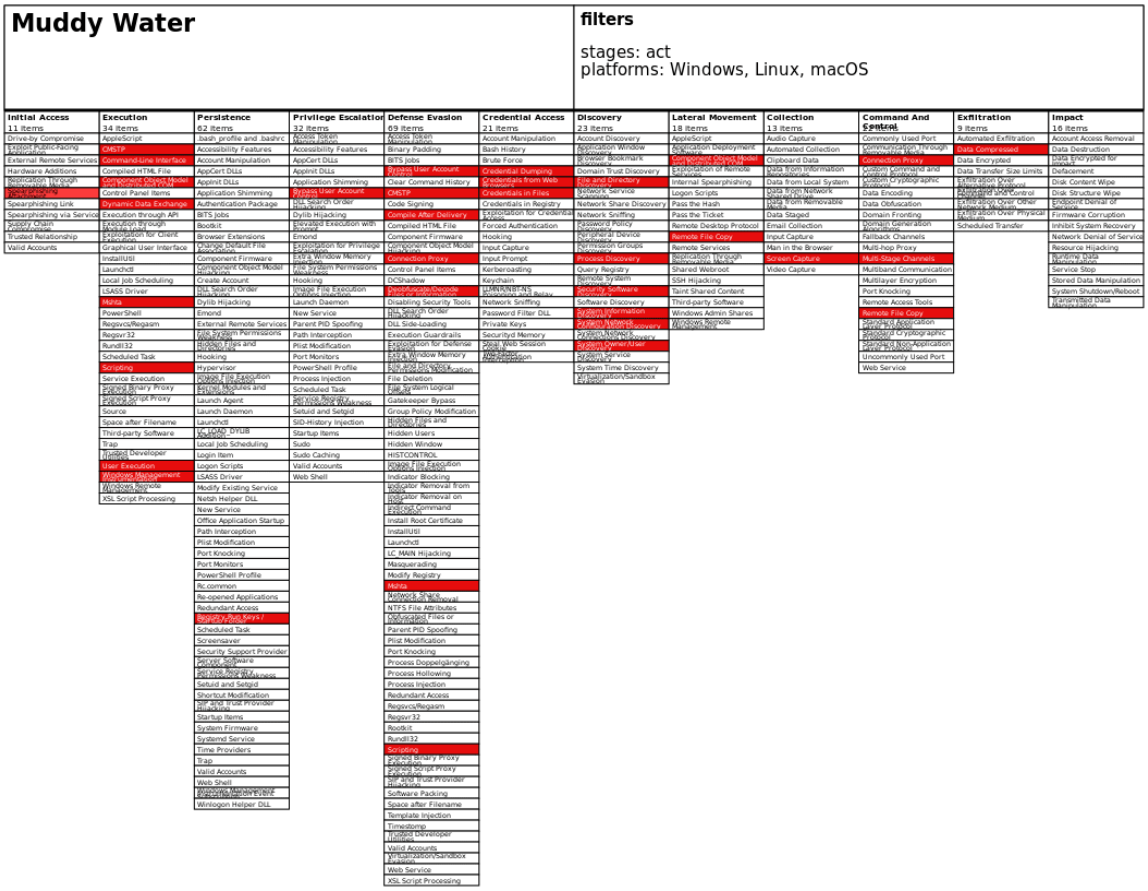
Let's suppose that security analysts receive a report about a new APT group that threats middle east and Africa. We can take "Muddy Water APT" as an example.

Go to https://mitre-attack.github.io/attack-navigator/enterprise/#

And highlight all the techniques used by Muddy Water APT Group



Export the techniques as SVG



If you are dealing with many APT groups at the same time highlight the techniques using colorful shades depends on how often the technique is used by the APT groups (brightest color = The technique is used by many groups)

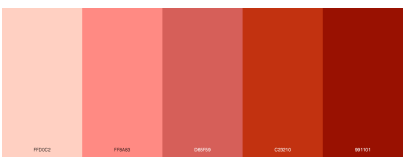
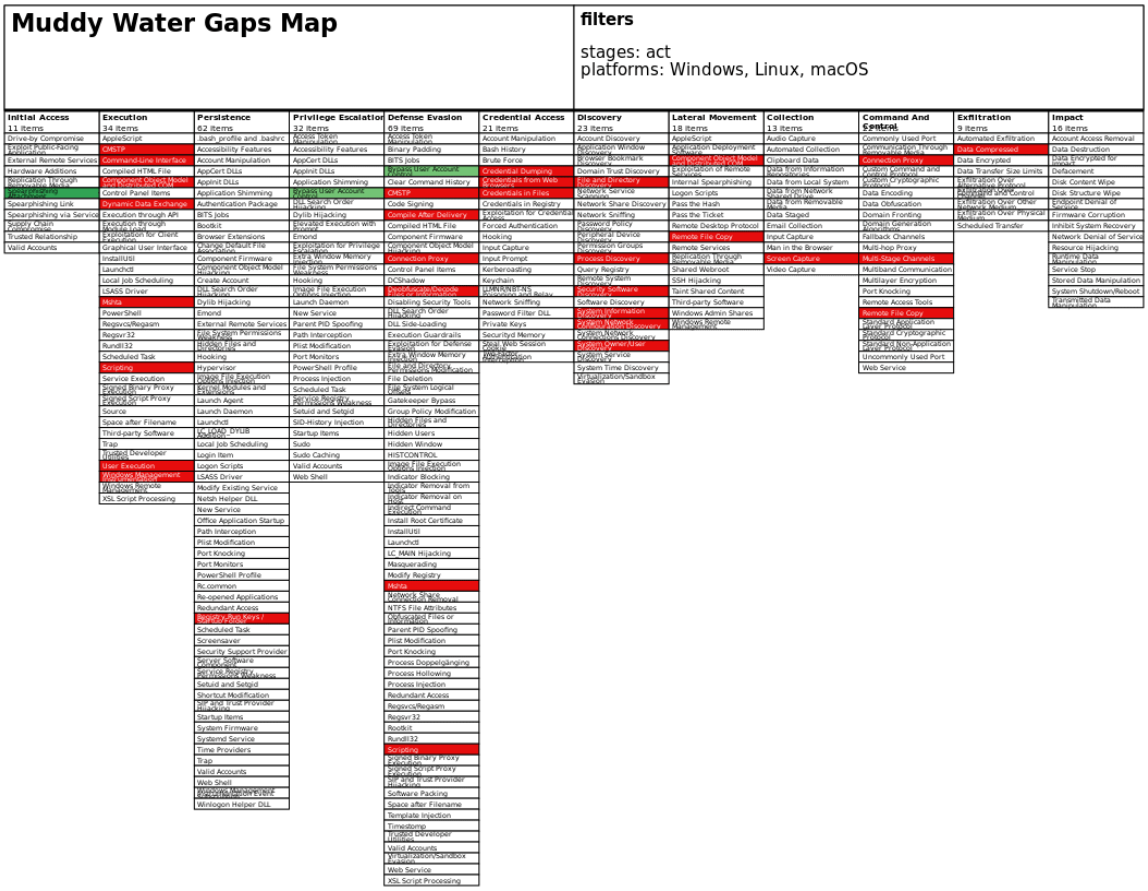


Image Courtesy_ _

Now you know your adversaries. It is time to prepare the mitigations (tools and techniques) and discover the gaps in our defenses.

Create a roadmap to improve the defense gaps and update the map accordingly



Mitigations for every technique can be found on https://attack.mitre.org/mitigations/enterprise/

Enterprise Mitigations

Mitigations: 41

ID	Name	Description
M1036	Account Use Policies	Configure features related to account use like login attempt lockouts, specific login times, etc.
M1015	Active Directory Configuration	Configure Active Directory to prevent use of certain techniques; use SID Filtering, etc.
M1049	Antivirus/Antimalware	Use signatures or heuristics to detect malicious software.
M1013	Application Developer Guidance	This mitigation describes any guidance or training given to developers of applications to avoid introducing security weaknesses that an adversary may be able to take advantage of.
M1048	Application Isolation and Sandboxing	Restrict execution of code to a virtual environment on or in transit to an endpoint system.
M1047	Audit	Perform audits or scans of systems, permissions, insecure software, insecure configurations, etc. to identify potential weaknesses.
M1040	Behavior Prevention on Endpoint	Use capabilities to prevent suspicious behavior patterns from occurring on endpoint systems. This could include suspicious process, file, API call, etc. behavior.

Summary

In this module, we learned many important terminologies and how to use MITRE ATT&CK framework to detect advanced persistent threats.

References

- https://www.fireeye.com/blog/products-and-services/2020/01/operationalizing-cti-hunt-for-defend-against-iranian-cyber-threats.html

Module 13 - Hands-on Malicious Traffic Analysis with Wireshark

Hands-on Malicious Traffic Analysis with Wireshark

Communication and networking are vital for every modern organization. Making sure that all the networks of the organization are secure is a key mission.In this article we are going to learn how to analyze malicious traffic using the powerful tool Wireshark.

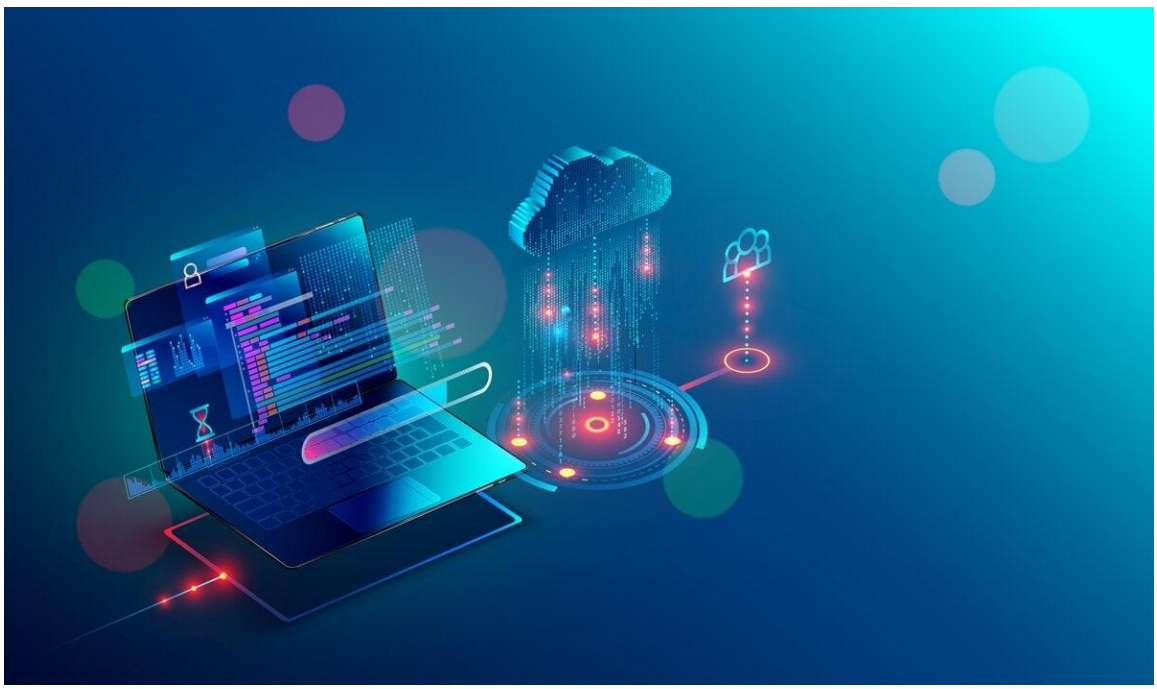


Image Courtesy

Before diving deep into traffic analysis, I believe that we need to explore some networking fundamentals first. It is essential to learn how a network works. Networking is the process of changing information between different devices. The transmission is usually done using a transmission mode. In communications we have generally 3 transmission modes:

- **Simplex Mode:** in this mode the data is transferred in one direction like the transmission used in TV broadcasting
- **Half-duplex Mode:** in this mode the data flows in two directions but using a single mean of communication
- **Full-duplex Mode:** in this mode the data flow is bidirectional and simultaneous.

When it comes to communication networks we have many types. Some of them are the following:

- **Local Area Network (LAN):** this network is used in small surfaces and areas
- **Metropolitan area network (MAN):** this network is larger than the Local Area Network. We can use for example to connect two offices.
- **Wide area network (WAN):** We use this type of networks to connect large distances
- **Personal area network (PAN):** this network is used in short distances and small areas like a single room.

Network Topologies

A topology is a schematic representation of a network. You can see it as the layout of the network and how the connected devices are arranged in the network. In networking we have many topologies some of the them are:

- **Ring Topology:** the data flows in one direction
- **Star Topology:** all the devices are connected to a single node (Hub)
- **Tree Topology:** this topology is hierarchical
- **Bus Topology:** all the devices are connected to a central connection
- **Fully-connected Topology:** each device is connected with all the other devices of the network

What is a network traffic?

Techopedia defines it as follows:

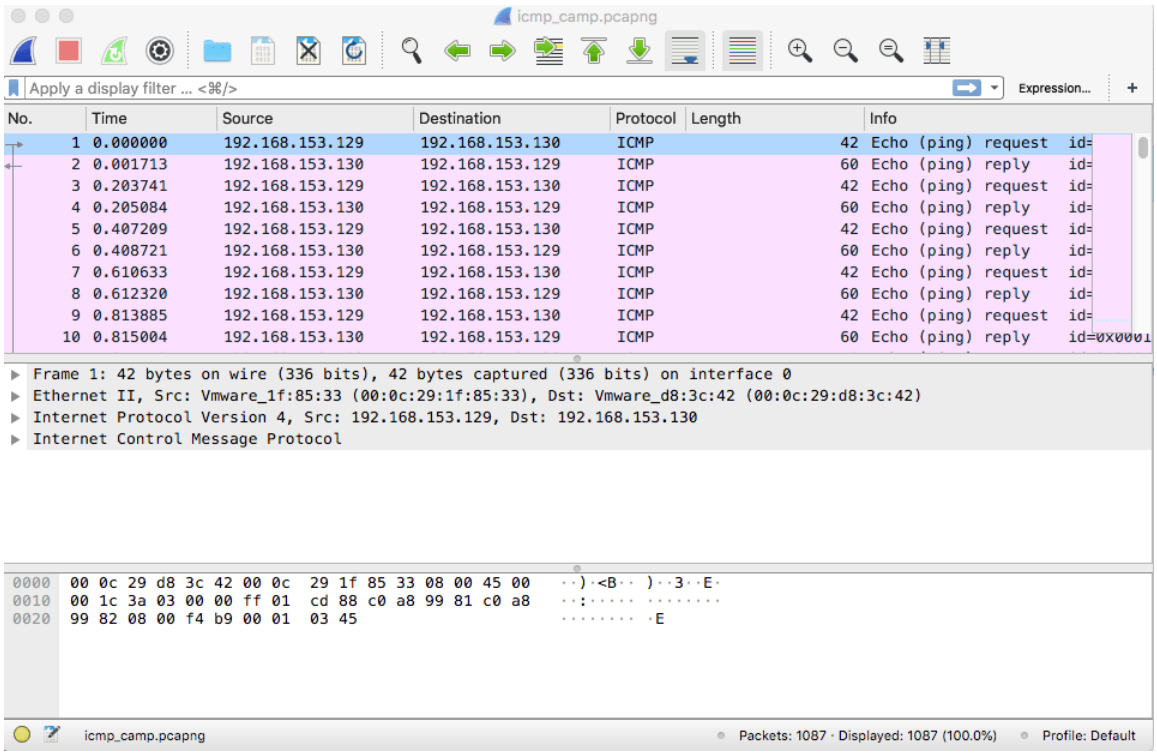
> *"Network traffic refers to the amount of data moving across a network at a given point of time. __Network data__ is mostly encapsulated in __network packets__ , which provide the load in the network. __Network traffic__ is the main component for network traffic measurement, network traffic __control__ and simulation."*

Image Courtesy

Traffic Analysis with Wireshark

The most suitable tool that will help you analyze your network traffic is definitely Wireshark. Wireshark is a free and open-source tool to help you analyse network protocols with deep inspection capabilities. It gives you the ability to perform live packet capturing or offline analysis. It supports many operating systems including Windows, Linux, MacOS, FreeBSD and many more systems.

You can download it from here: https://www.wireshark.org/download.html



Wireshark will help capture and analyze traffic as **pcap** files. The analysis follows the OSCAR methodology:

- Obtain
- Strategize
- Collect Evidence
- Analyze
- Report

Image Courtesy

Let's start by analyzing a sample pcap file so we can understand Wireshark capabilities. But before that we need to know an important model called the **OSI netwoking Model** :

By Definition: "The **Open Systems Interconnection model** (**OSI model**) is a conceptual model that characterizes and standardizes the communication functions of a telecommunication or computing system without regard to its underlying internal structure and technology. Its goal is the interoperability of diverse communication systems with standard protocols. The model partitions a communication system into abstraction layers. The original version of the model defined seven layers.

In other words data is moving in the network respecting a specific order. The following are the seven Layers of the OSI Model:

7- Application layer

6 -Presentation layer

5- Session layer

4- Transport layer

3- Network layer

2- Data link layer

1- Physical layer

The following graph illustrates the different OSI model layers:

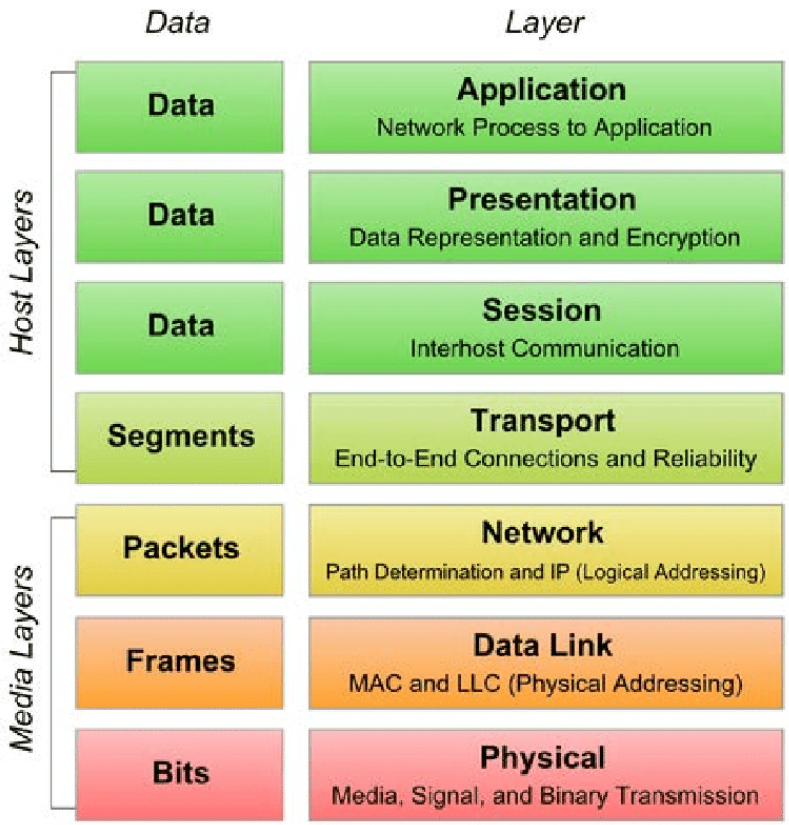
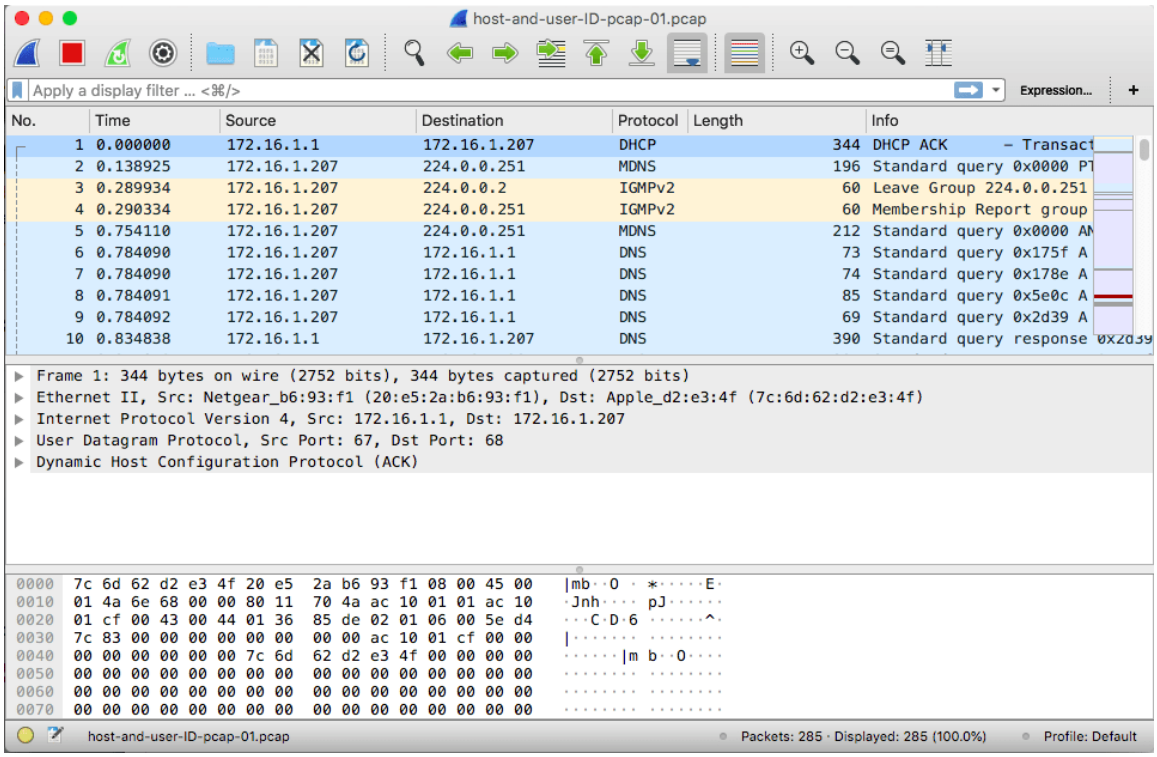


Image Courtesy

As a first demonstration let's start analyze a small pcap delivered by *malware-traffic-analysis.net. _The file password is "_infected*"

Once you open it with Wireshark you will get this main window:

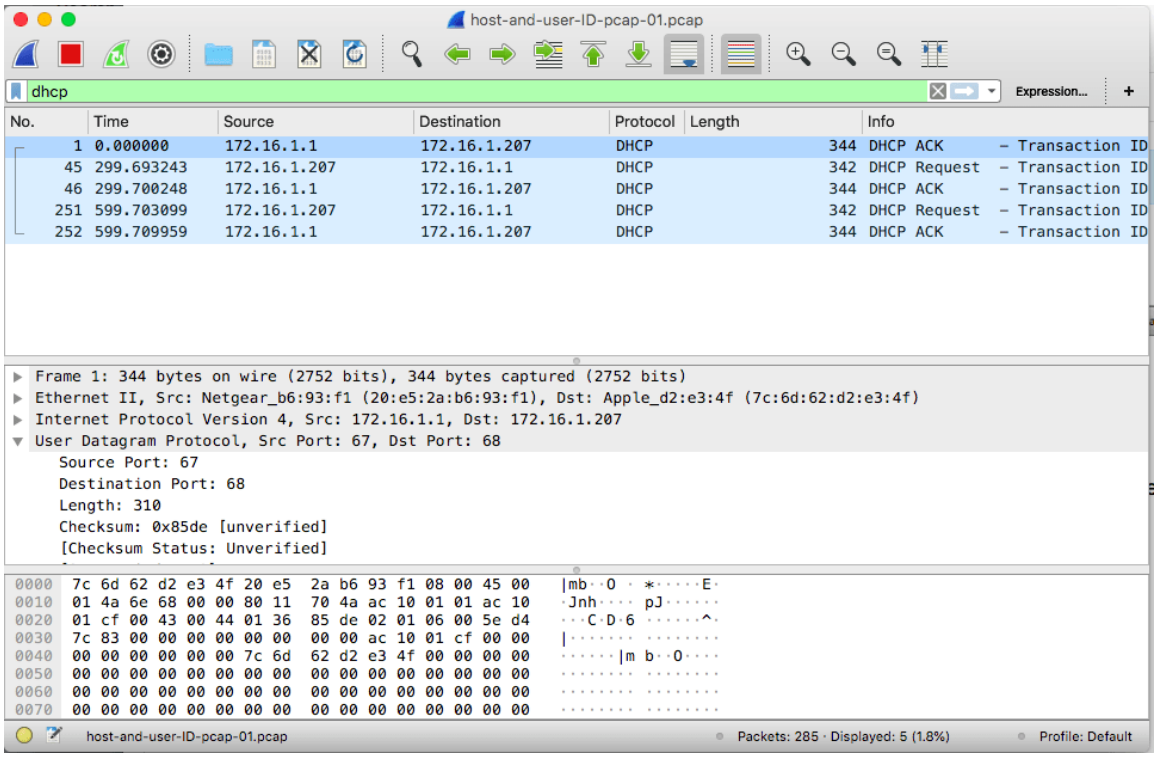


Let's start collecting some helpful information like the Host, destination, source etc...

To get the host we can use the DHCP filter.

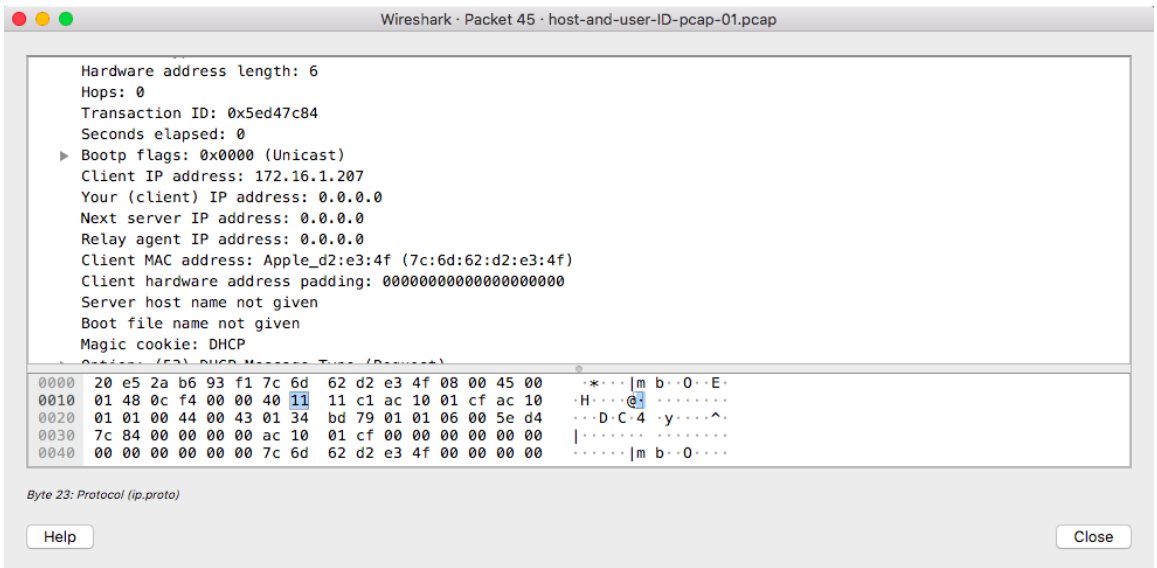
Dynamic Host Configuration Protocol (DHCP) is a network layer protocol based on RFC 2131 that enables assigning IP addresses dynamically to hosts. It goes through 4 steps:

- _Discovery _
- *Offer*
- *Request*
- *Acknowledgment*

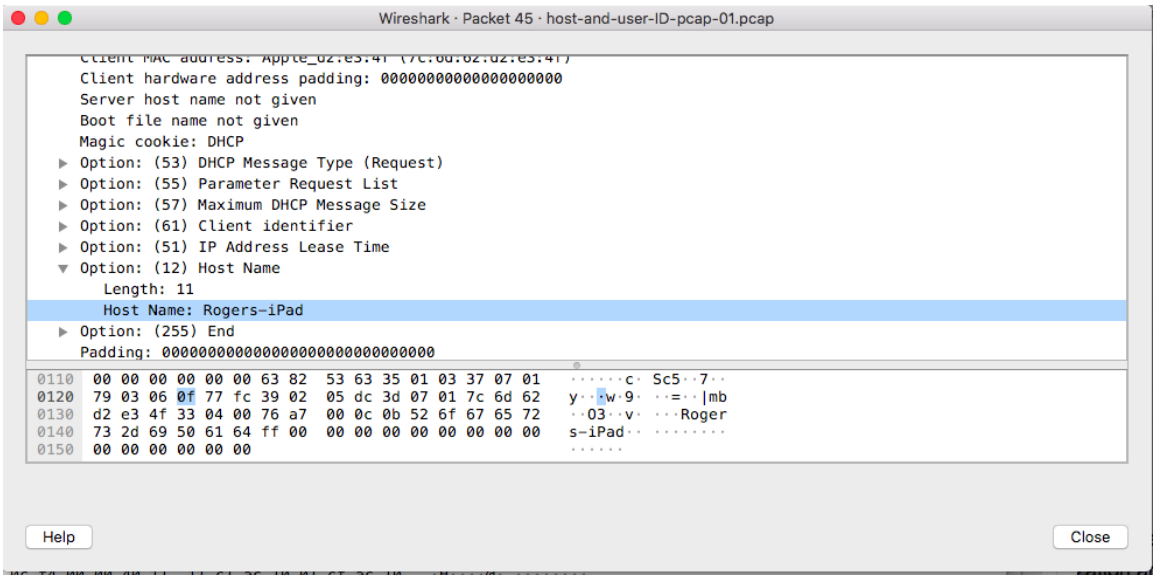


To learn more about Filters check this great resource: Using Wireshark – Display Filter Expressions

Now select: DHCP Request and you will get many helpful pieces of information including the client Mac address. In switching the traffic of data is determined by Media Access Control (MAC) addresses. A MAC address is a unique 48-bit serial number. It is composed equally of the Organizational Unique Identifier (OUI) and the vendor-assigned address.MAC addresses are stored in a fixed size table called the Content Addressable Memory (CAM)



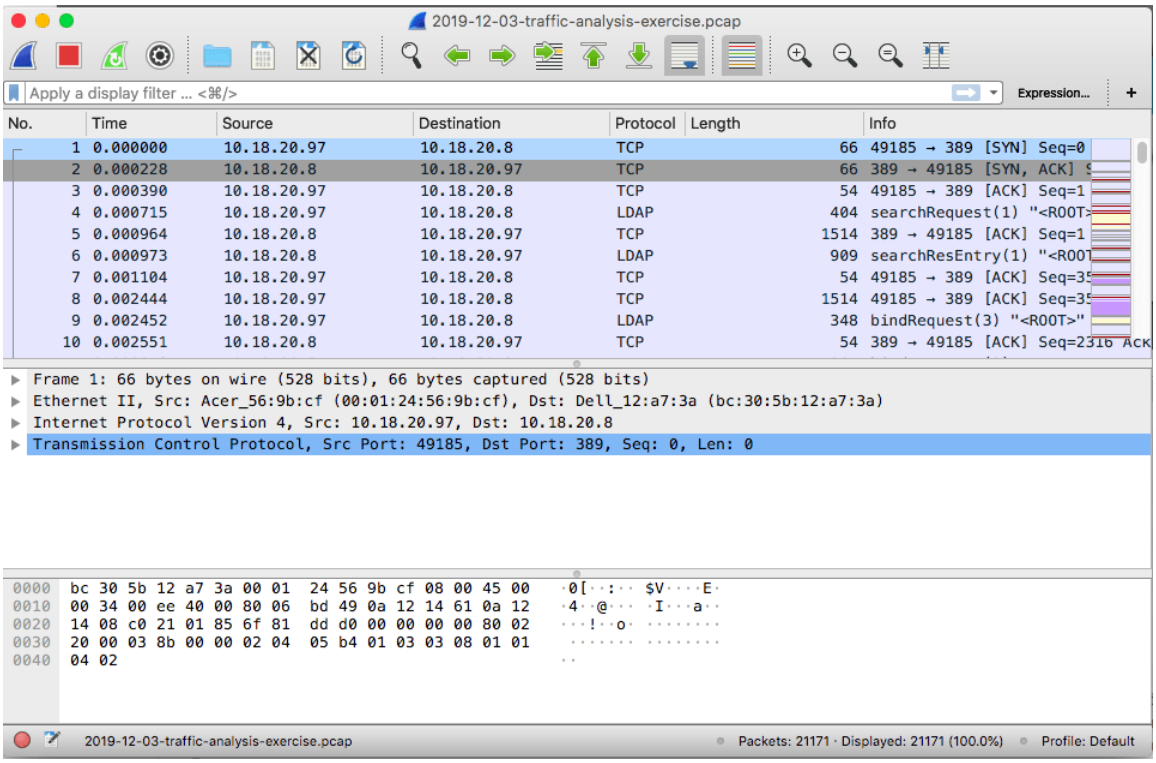
And you will get also the hostname. It is "*Rogers-iPad*"



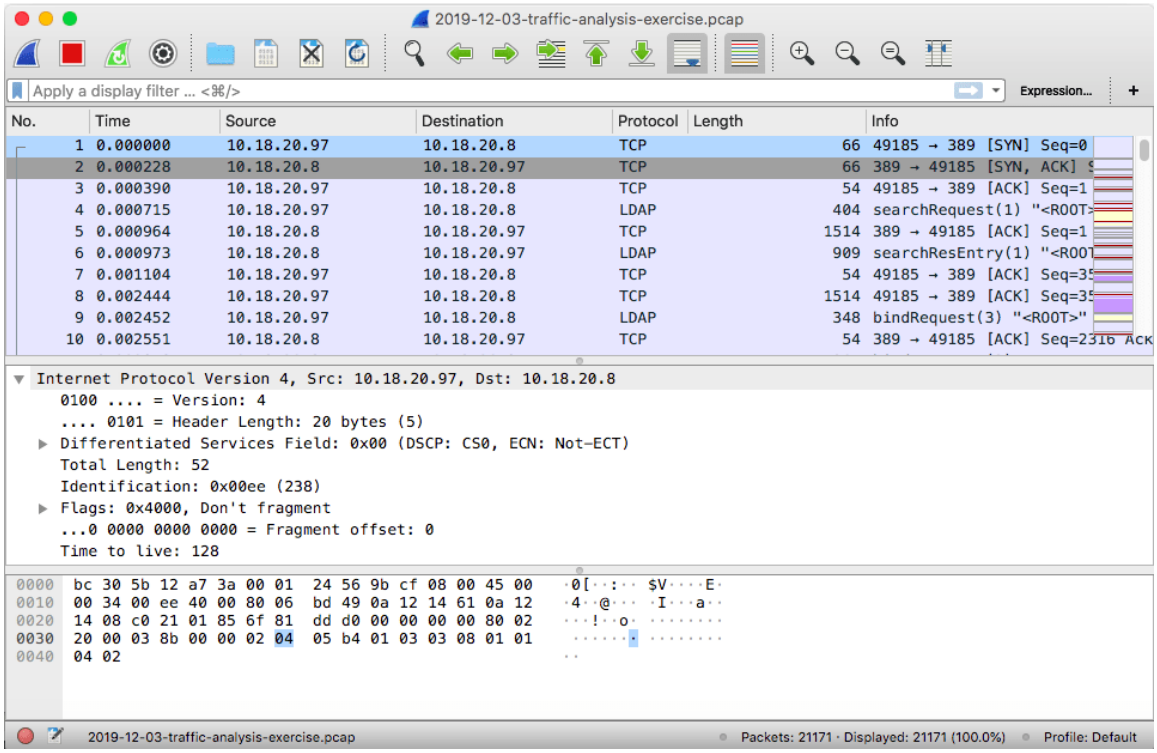
After taking a look at how you can use Wireshark to extract some pieces of information, let's analyze a malicious traffic. As a demonstration we are going to analyze this pcap from the same source (the password is "infected"). Some additional alerts file can be found here.

Open the pcap file with Wireshark. We are going to find:

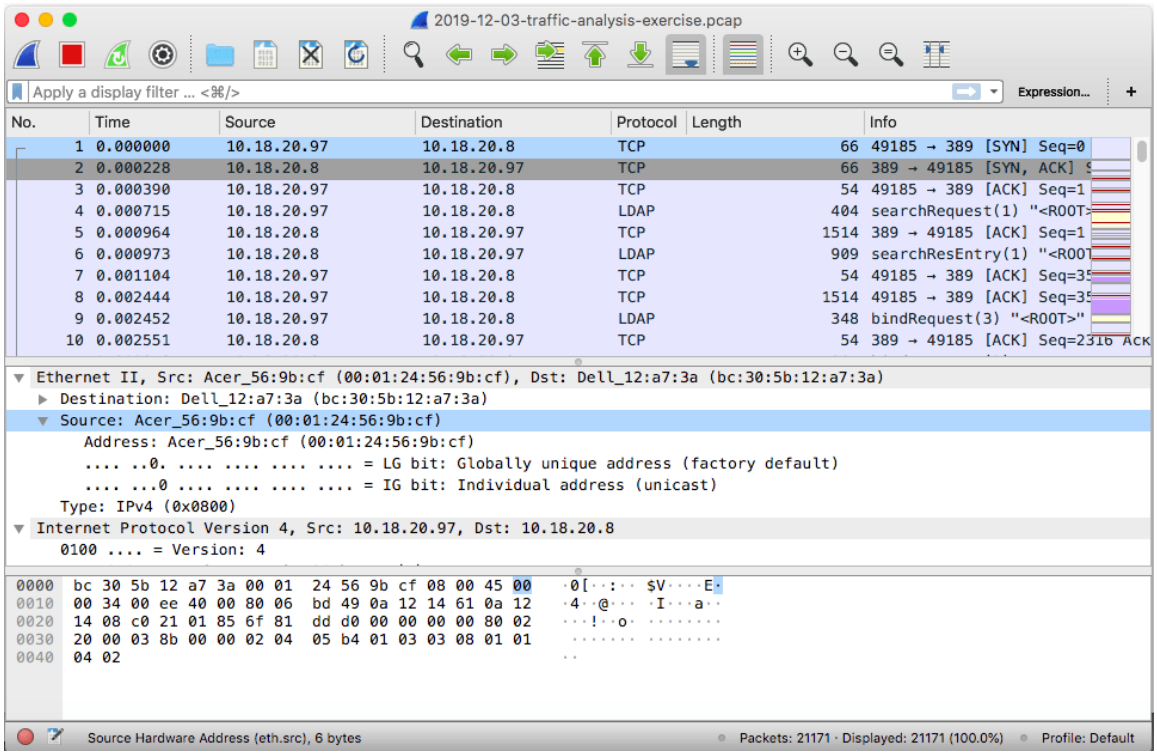
- The IP address, MAC address, and host name of the infected Windows host
- The Windows user account name of the victim
- The used Malware



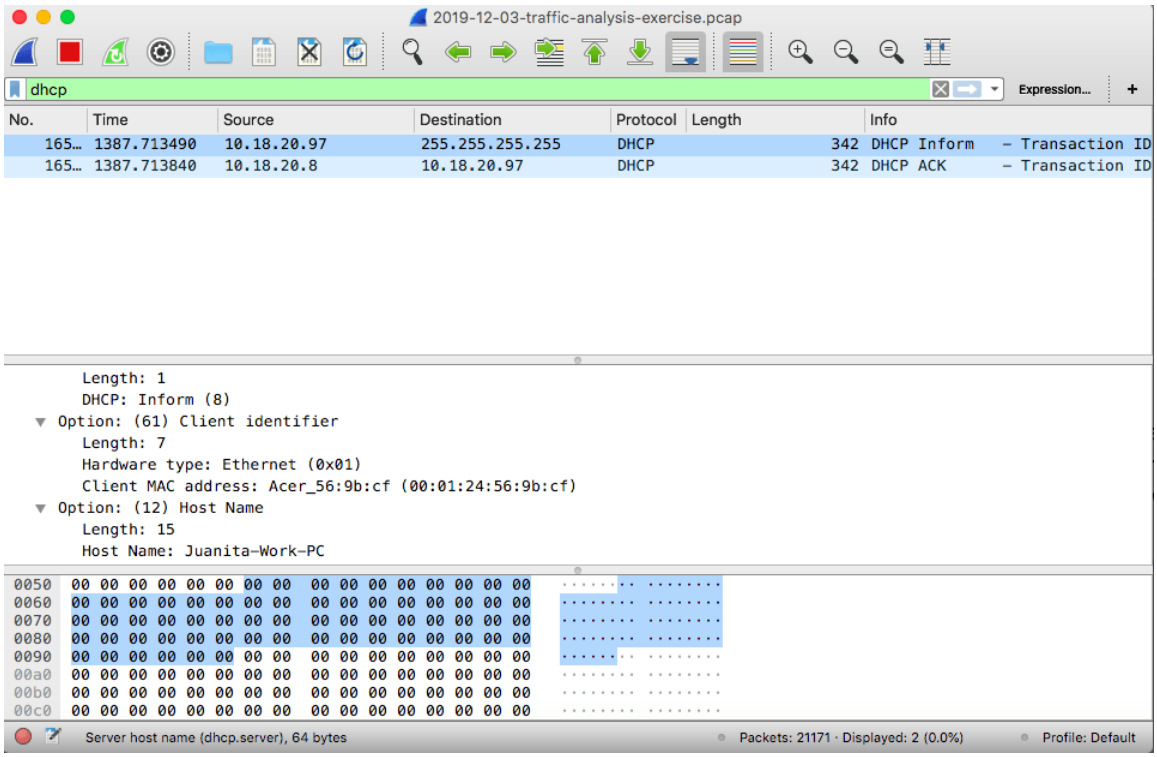
By highlighting "Internet Protocol Version 4" we can get the IP address which is: **10.18.20.97**



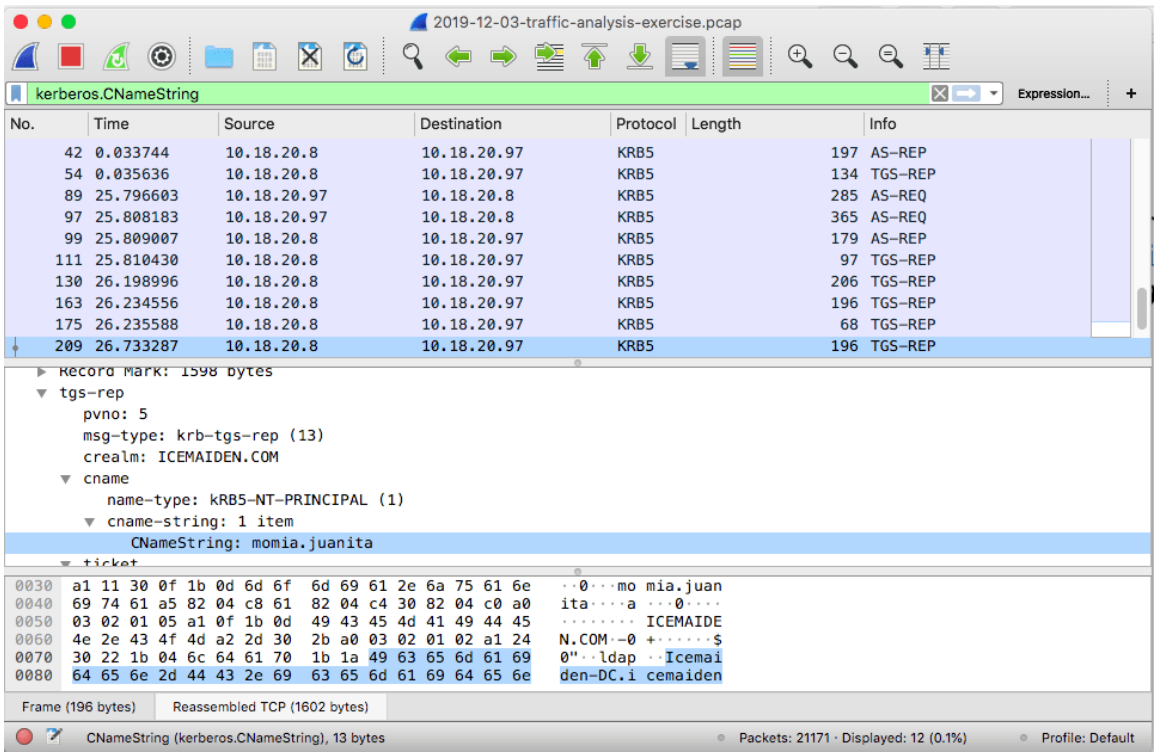
The MAC address is: **00:01:24:56:9b:cf**



Like what we did previously to detect the hostname we can see that the hostname is: JUANITA-WORK-PC

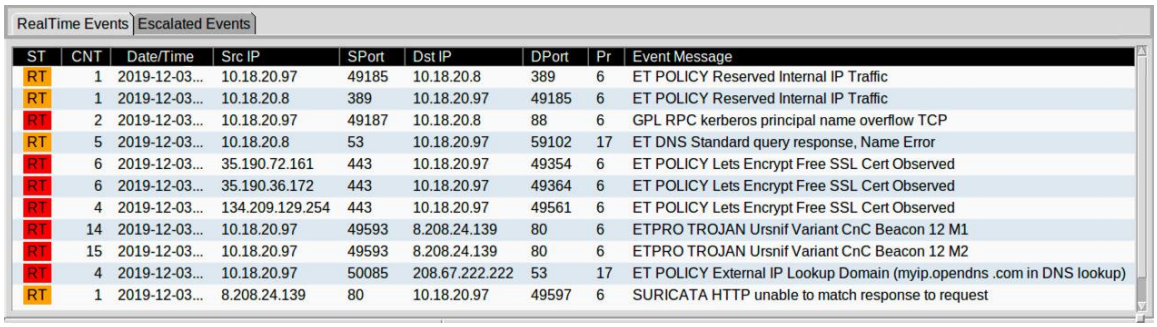


To get the windows user account by analyzing the kerberos traffic using this filter: _ **kerberos.CNameString** _



The Windows account name is: **momia.juanita**

Based on the alerts we can get that the malware was a variant of " **Ursnif**"

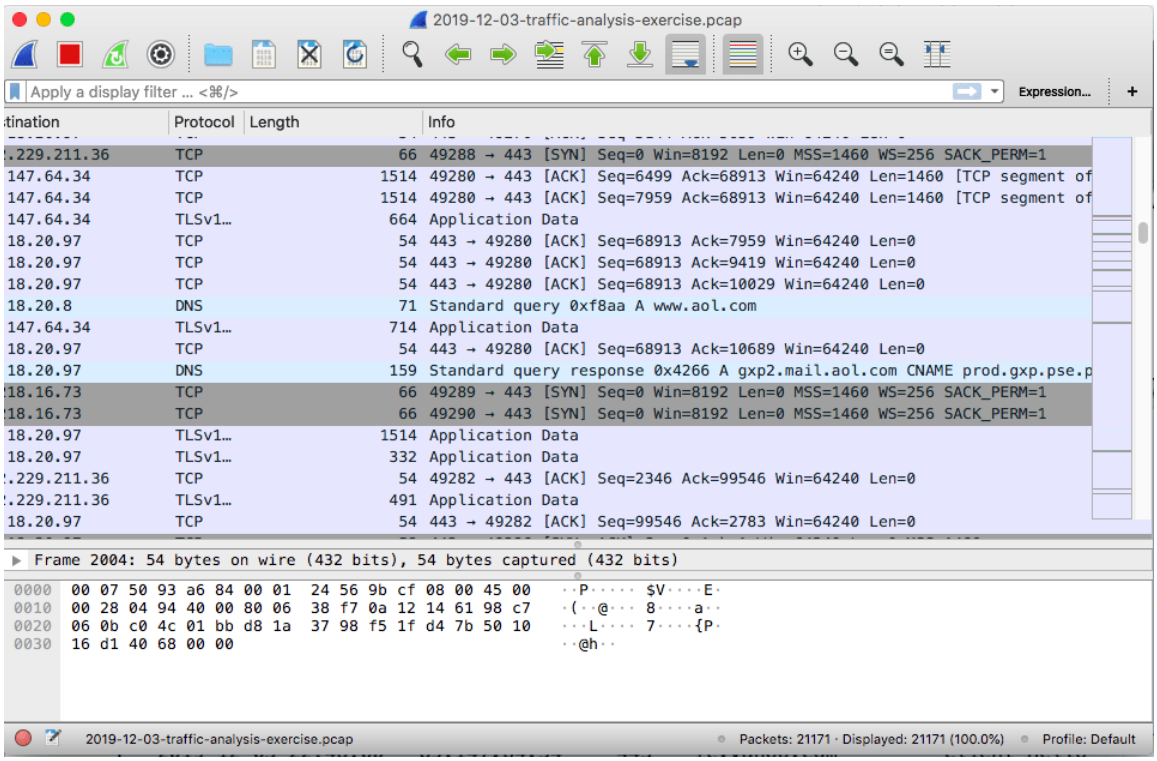


RealTime Events | Escalated Events

ST	CNT	Date/Time	Src IP	SPort	Dst IP	DPort	Pr	Event Message
RT	1	2019-12-03...	10.18.20.97	49185	10.18.20.8	389	6	ET POLICY Reserved Internal IP Traffic
RT	1	2019-12-03...	10.18.20.8	389	10.18.20.97	49185	6	ET POLICY Reserved Internal IP Traffic
RT	2	2019-12-03...	10.18.20.97	49187	10.18.20.8	88	6	GPL RPC kerberos principal name overflow TCP
RT	5	2019-12-03...	10.18.20.8	53	10.18.20.97	59102	17	ET DNS Standard query response, Name Error
RT	6	2019-12-03...	35.190.72.161	443	10.18.20.97	49354	6	ET POLICY Lets Encrypt Free SSL Cert Observed
RT	6	2019-12-03...	35.190.36.172	443	10.18.20.97	49364	6	ET POLICY Lets Encrypt Free SSL Cert Observed
RT	4	2019-12-03...	134.209.129.254	443	10.18.20.97	49561	6	ET POLICY Lets Encrypt Free SSL Cert Observed
RT	14	2019-12-03...	10.18.20.97	49593	8.208.24.139	80	6	ETPRO TROJAN Ursnif Variant CnC Beacon 12 M1
RT	15	2019-12-03...	10.18.20.97	49593	8.208.24.139	80	6	ETPRO TROJAN Ursnif Variant CnC Beacon 12 M2
RT	4	2019-12-03...	10.18.20.97	50085	208.67.222.222	53	17	ET POLICY External IP Lookup Domain (myip.opendns .com in DNS lookup)
RT	1	2019-12-03...	8.208.24.139	80	10.18.20.97	49597	6	SURICATA HTTP unable to match response to request

_ **Ursnif** steals **system information and attempts to** steal banking__ and online account credentials. (from: F-Secure **Labs**: *https://www.f-secure.com/v-descs/trojan_w32_ursnif.shtml*)_

The malware appears to come from a mail because if you notice closely you will find that the victim visited mail.aol.com:



I hope you found it helpful.

Summary

In this article, we explored Wireshark and how to use to perform malicious traffic analysis.

To learn more about traffic analysis you can download this doc that contains many useful resources: Malicious Traffic Analysis Resources

References and Credit

- https://unit42.paloaltonetworks.com/using-wireshark-identifying-hosts-and-users/
- https://www.malware-traffic-analysis.net/2019/12/03/index.html

Hands-on Guide to Digital Forensics

Digital forensics is one of the most interesting fields in information security. In this post, we will explore what digital forensics is and we will learn how to perform some digital forensics tasks using some powerful tools and utilities.

In this article we are going to explore the following points:

- **Digital Forensics Fundamentals**
- **Digital Forensics Lab**
- **Network evidence collection and Analysis**
- **Host-based evidence collection and Analysis**
- **Forensics Imaging**
- **Practical Lab: Autopsy Forensics Browser**
- **Practical Lab: Memory Analysis with Volatility**

Digital Forensics Fundamentals

Before diving into the practical labs it is essential to explain many important terminologies. First, *what is digital forensics*?

NIST is describing Forensics as the following:

> _The most common goal of performing forensics is to gain a better understanding of an event of interest by finding and analyzing the facts related to that event... Forensics may be needed in many different situations, such as evidence collection for legal proceedings and internal disciplinary actions, and handling of malware incidents and unusual operational problems. _

Like any methodological operation, Computer forensic analysis goes through well-defined steps: **Identification** , **Preservation** , **Collection** , **Examination** , **Analysis** and **Presentation**.

Figure

let's explore these steps one by one:

- Identification
- Preservation
- Collection
- Examination
- Analysis
- Presentation

According to worldsecuresystems:

> "A chain of custody is a document that is borrowed from law enforcement that tracks evidence from the time the Computer Forensics Examiner gains possession of the item until it is released back to the owner. "

The following illustration presents a chain of custody template:

CHAIN OF CUSTODY FORM

Your Logo Here		Your Address Here

[Agency Name] Case #:	

Item #	Date/Time Removed	Reason for Removal of Evidence	Signature

Figure

Digital Forensics Lab

To perform digital forensics, obviously, you need to prepare a lab for it. It is essential to have both the required hardware and software.

Hardware

During investigations, digital forensics experts are dealing with many hardware pieces and devices including RAMs and Storagemedia devices. Thus, it is important to acquire a suitable hardware equipment to perform the task in good condition. Some of the required hardware pieces are the following:

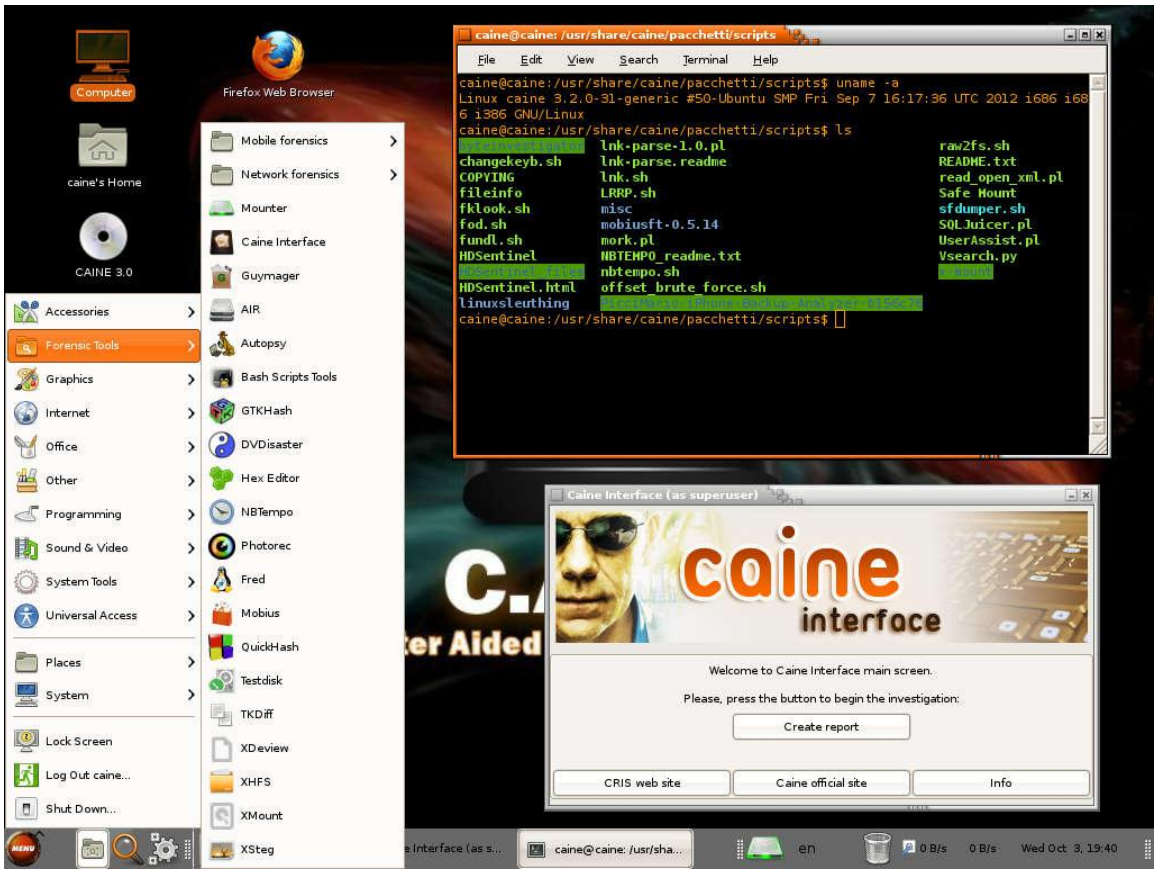
- A digital Forensics laptop (A minimum of 32 GB of RAM is recommended) with an OS that contains the needed digital forensics tools
- A secondary machine with Internet connexion
- A physical write blocker

Figure

Software

As I said previously, a digital forensics computer needs to be equipped with many DF tools. Some of the most used tools and operating systems are the following:

- *SANS SIFT*
- *CAINE OS*
- _Volatility _
- *X-Ways Forensics*
- *Autopsy: the Sleuth Kit*
- *Bulk Extractor*



Figure

Network evidence collection and Analysis

An evidence is the information to be investigated. Digital forensics analysts are dealing with different categories of evidence including **network-based evidence** and **host-based evidence**. Let's start exploring how to deal with network evidence. As we cited earlier, the first step is collecting the evidence. In networking, we can perform the collection using many techniques and tools. After identifying the source of evidence using for example network diagrams, you can use packet capture tools such as:

TCPdump

"Tcpdump is a powerful command-line packet analyzer; and __libpcap__ , a portable C/C++ library for __network traffic__ capture." (Source: tcpdump.org)

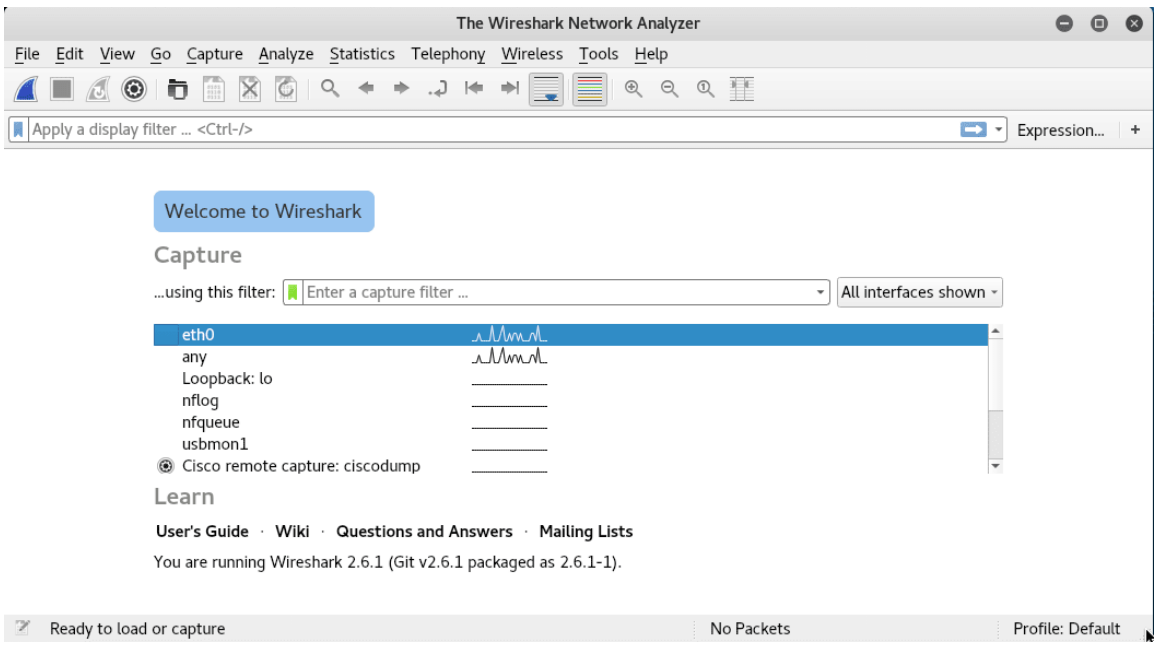
Wireshark

> *"Wireshark is the world's foremost and widely-used __network protocol__ analyzer. It lets you see what's happening on your network at a microscopic level and is the de facto (and often de jure) __standard__ across many commercial and non-profit __enterprises__ , __government agencies__ , and educational institutions. __Wireshark__ __development__ thrives thanks to the __volunteer__ contributions of networking __experts__ around the globe and is a continuation of a __project__ started by Gerald Combs in 1998". (Source: __**wireshark.org**__)*

As a demonstration let's explore how to analyse a small **pcap** file with Wireshark.

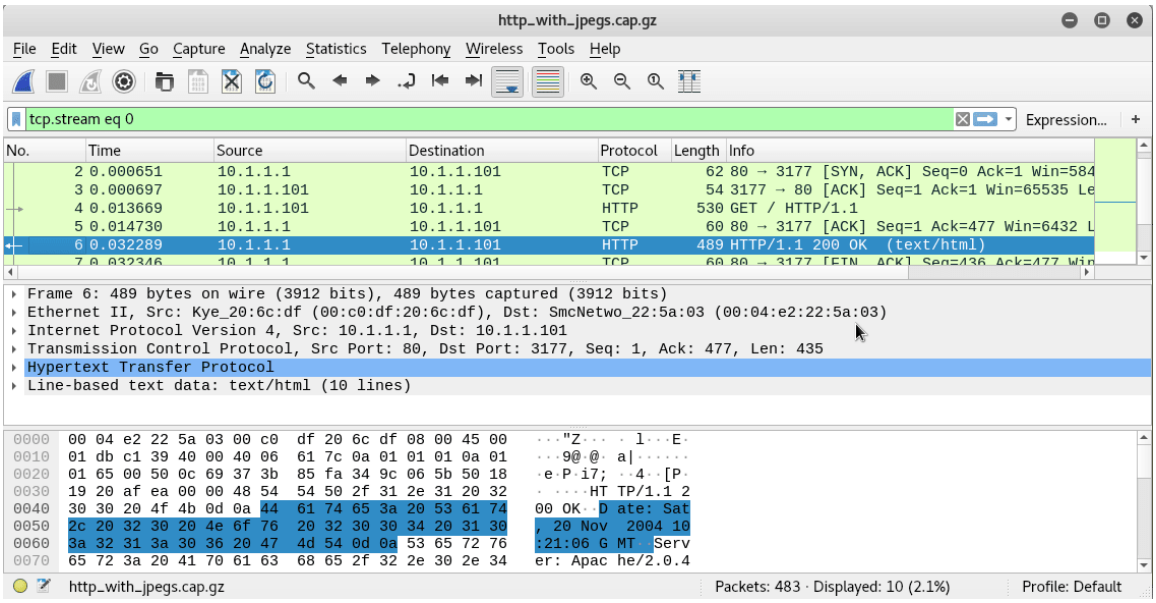
If you are using Kali Linux, Wireshark is already installed there.

Open Wireshark

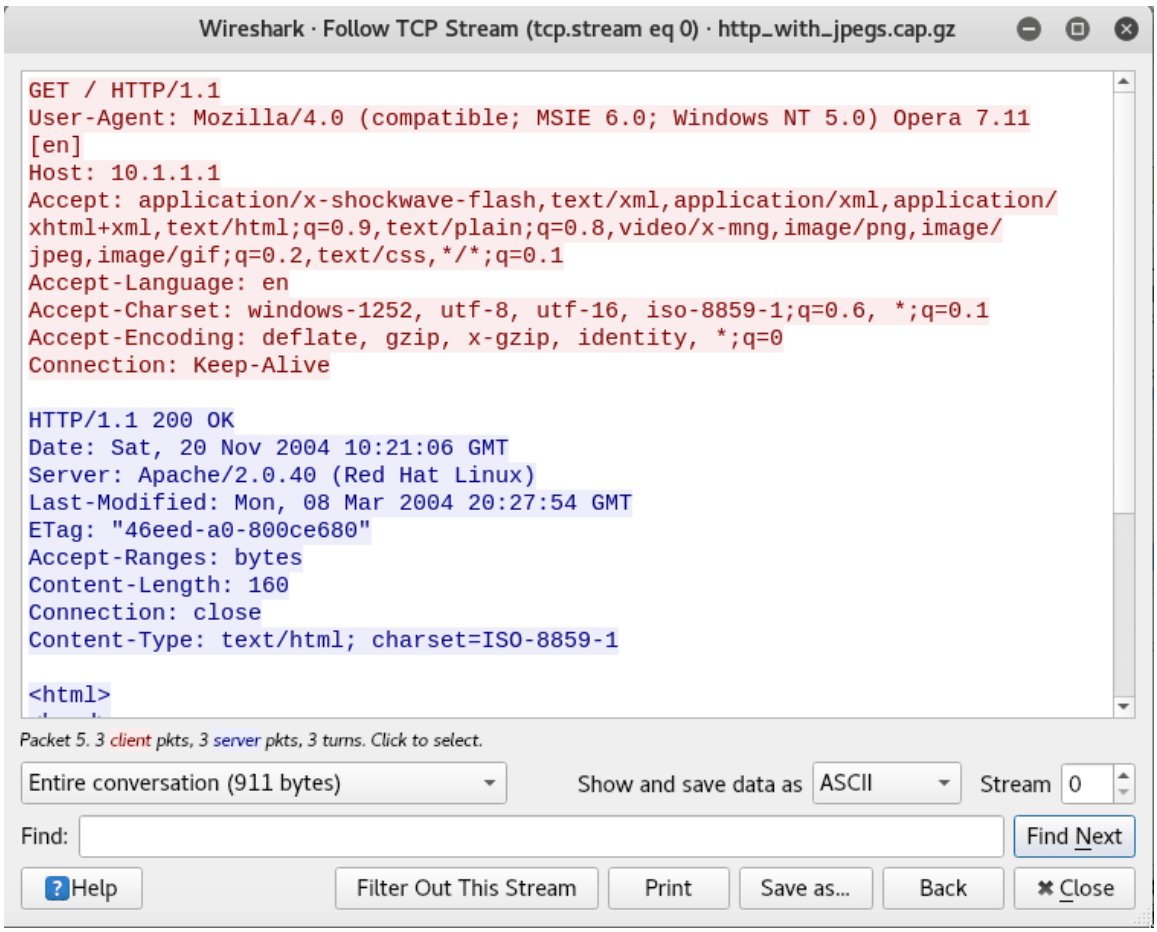


We are going to analyse this Pcap file http_with_jpegs.cap.gz from here: https://wiki.wireshark.org/SampleCaptures

Open the file with Wireshark:

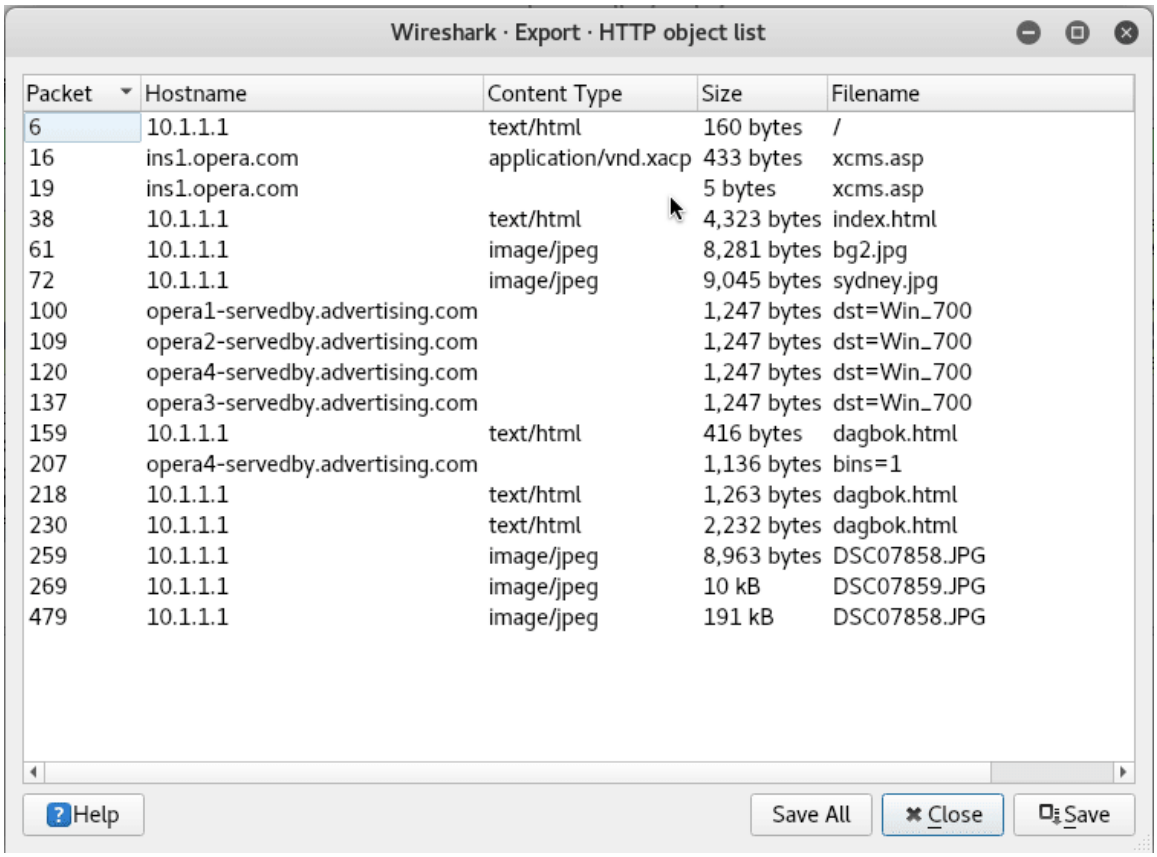


To select a TCP stream go to **Analyze** -\> **follow TCP stream**

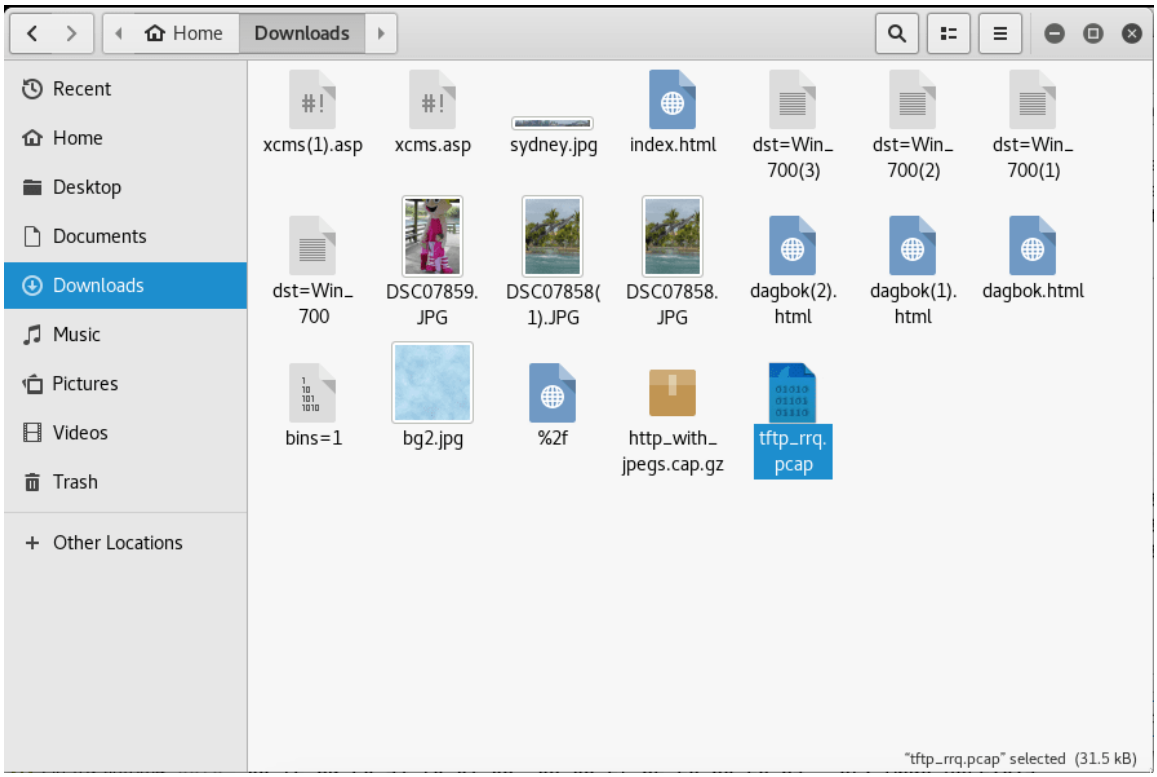


For example, we are going to extract the files from the captured packet:

Go to **File** -\> **Export objects** -\> **HTTP** -\> **Save all**



Voila! we extracted the included files:



Host-based evidence collection and Analysis

As an investigator and digital forensics expert, it is essential to acquire knowledge about the different storage means and the different filesystems. By definition, a storage media is a device where we can store data and information. There are many used storage devices including:

- *Hard drive*
- *DVD-ROM*
- *USB drive*
- _Memory cards and so on _

Figure

The removable storage media pieces need to be formatted with a specific filesystem. Some of the most used filesystems are:

- *Ext4*
- *Ext3*
- *NTFS*
- *FAT32*

To collect host-based evidence, you need to know the difference between volatile data and non-volatile data. Volatile data is data that is lost after a shutdown or some system changes. CPU data and ARP cache are some forms of volatile data. Data stored in hard drives and Master File Table (MFT) entries are non-volatile data. The host-based evidence acquisition can be done locally or remotely. Also, it can be done online or offline. Evidence collection is performed with what we call "Forensics Imaging"

Forensics Imaging

Forensics imaging is a very important task in digital forensics. Imaging is copying the data carefully with ensuring its integrity and without leaving out a file because it is very critical to protect the evidence and make sure that it is properly handled. That is why there is a difference between normal file copying and imaging. Imaging is capturing the entire drive. When imaging the drive, the analyst image the entire physical volume including the **master boot record**. There are two imaging techniques:

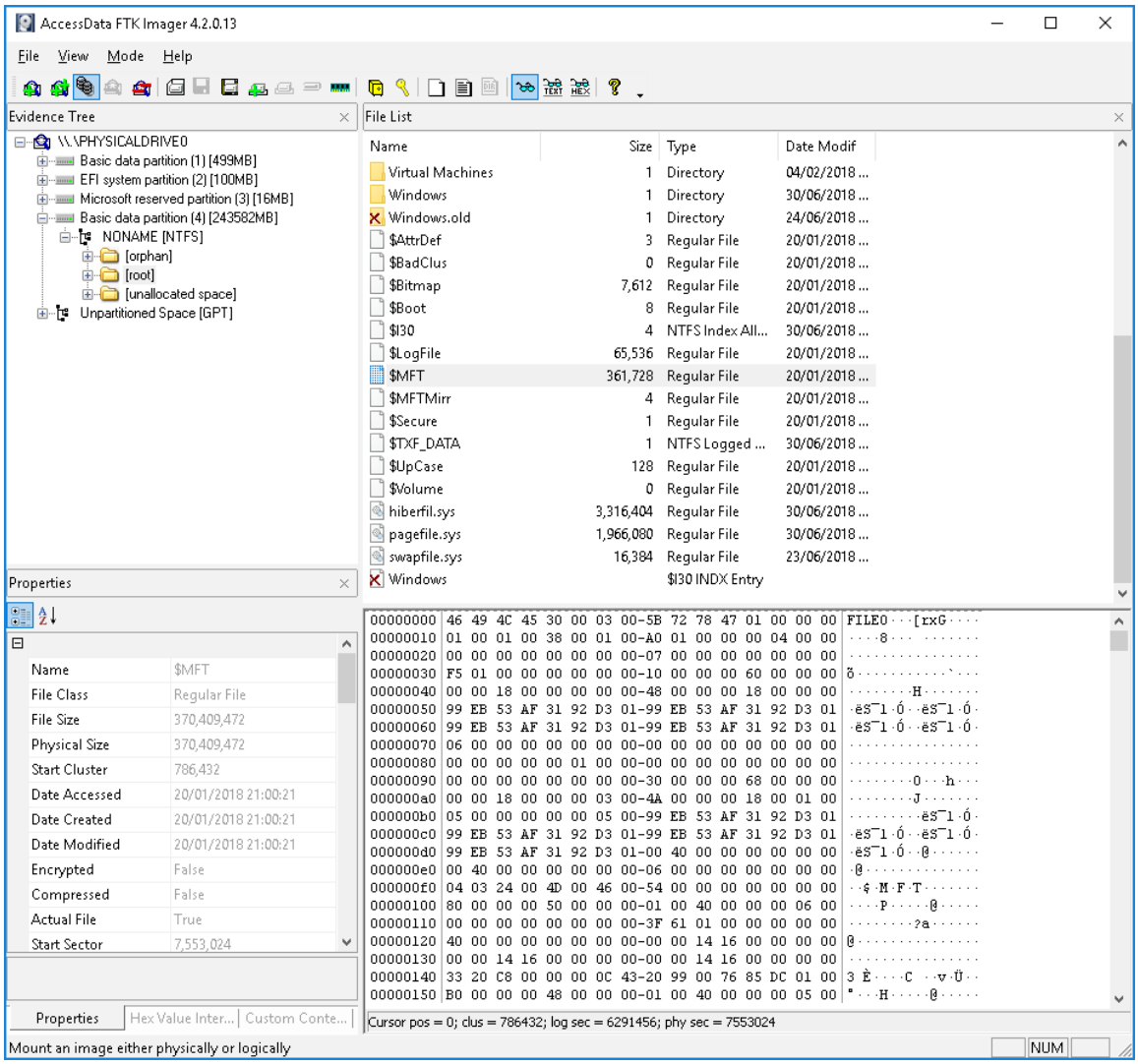
- Live imaging: the compromised system is not-offline
- Dead imaging: the compromised system is offline

Also, the taken images can be in many formats such as:

- Raw images
- EnCase evidence files
- AFF
- Smart and so on

For imaging, you can use FTK Imager:

"*FTK Imager is a data preview and imaging __tool__ used to acquire data (evidence) in a __forensically__ sound manner by creating copies of data without making changes to the original evidence.*"



Figure

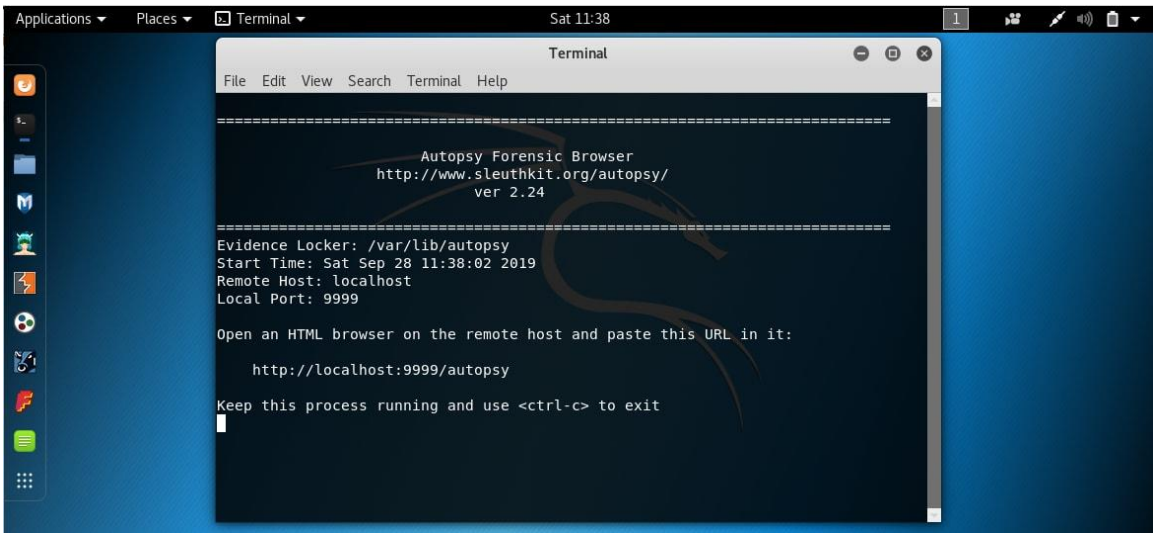
Practical Lab 1: Autopsy Forensics Browser

As a second demonstration, we are going to learn how to use a great forensics tool called "Autopsy Forensics Browser". According to https://www.linuxlinks.com/autopsy/ :

> *The Autopsy __Forensic__ __Browser__ is a graphical __interface__ to the __command line__ digital __investigation__ tools in The Sleuth Kit. The two together enable __users__ to investigate volumes and file __systems__ including __NTFS__ , __FAT__ , UFS1/2, and Ext2/3 in a 'File Manager' style interface and perform __key__ __word__ searches.*

If you are using Kali Linux, can found it directly there without the need to install it:

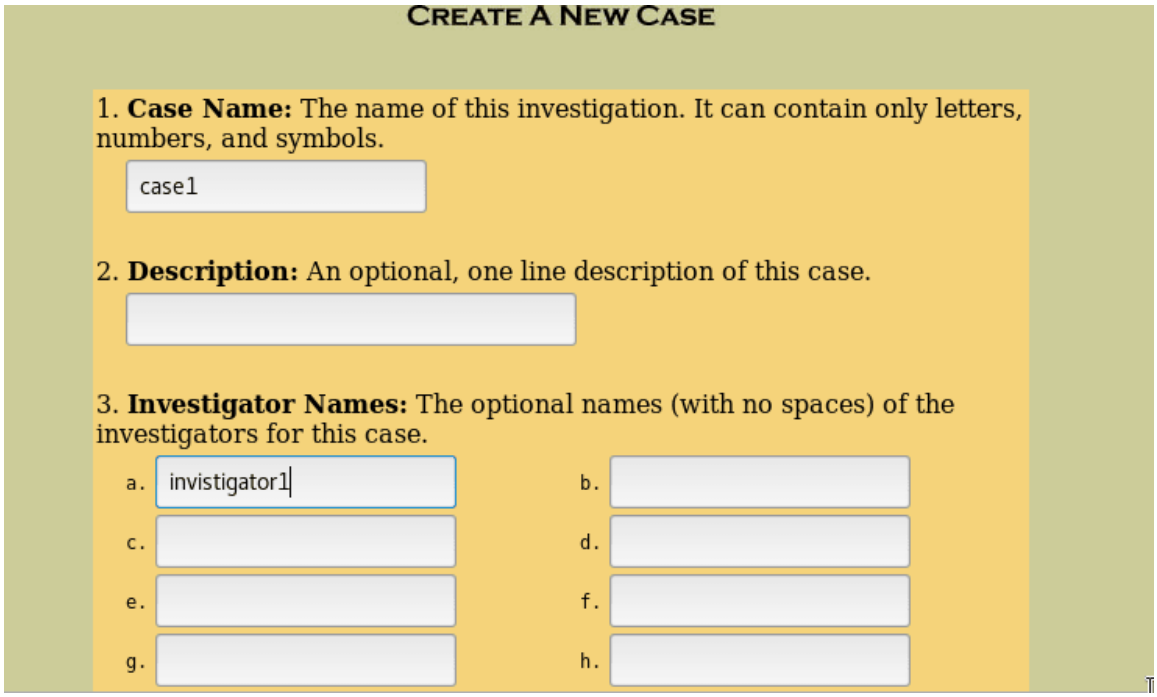
Run it from the menu:



Go to:

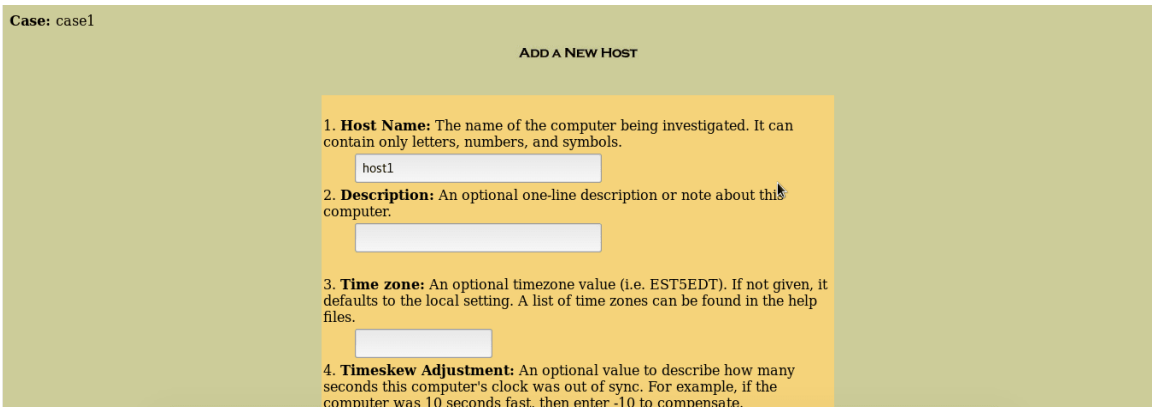
`http://localhost:9999/autopsy`

Create a new case:



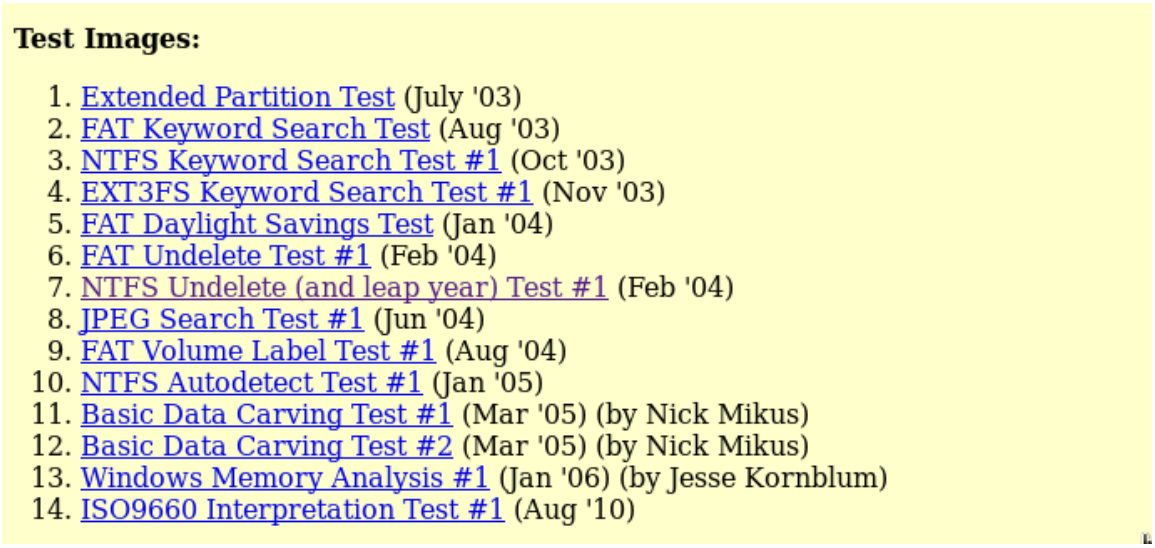
Select the profile

Add a host

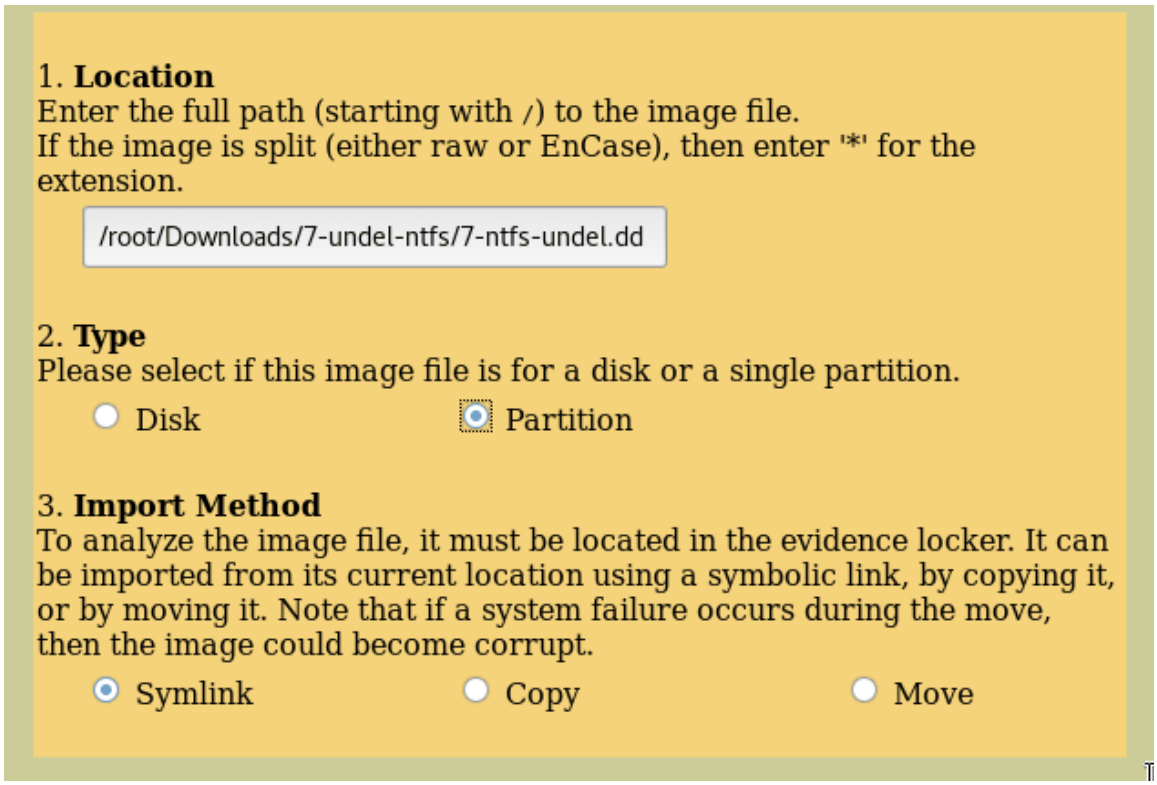


Check the configuration and click Add Image

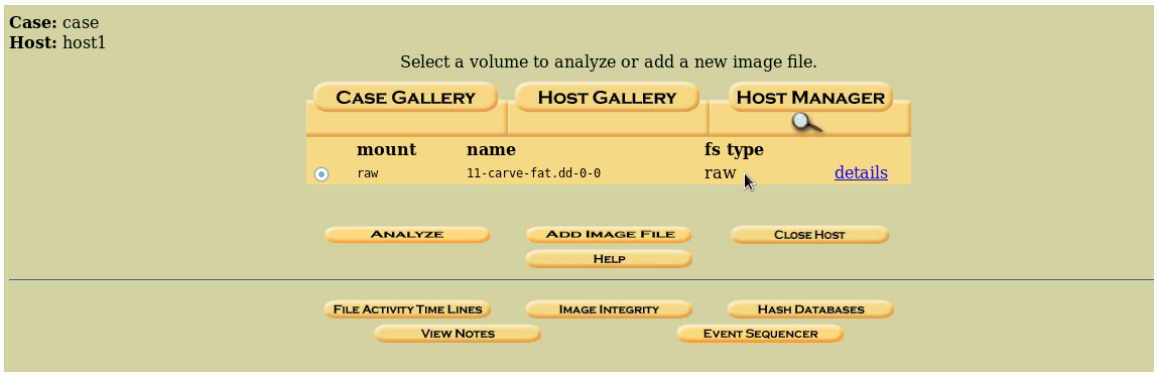
For the demo, we are going to use a memory dump sample (NTFS Undelete) from http://dftt.sourceforge.net (Digital Forensics Tool Testing Images)



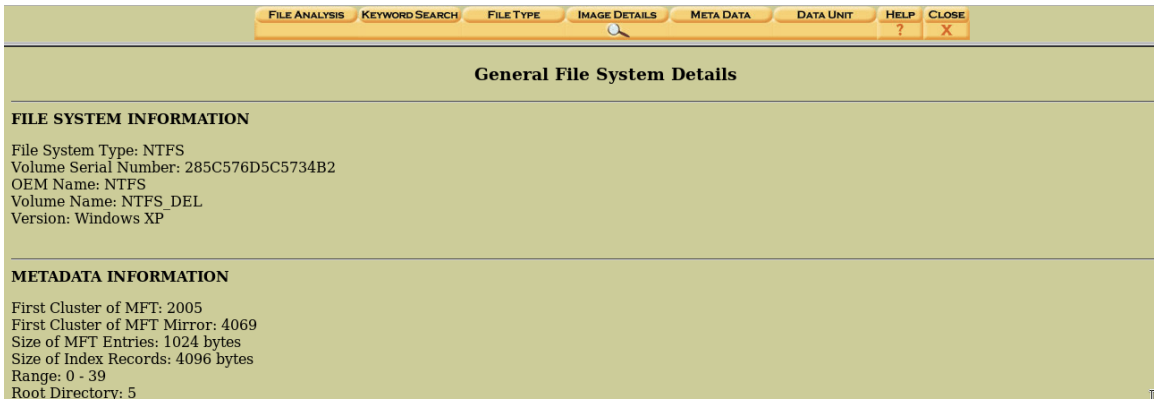
Add the path of the dump:



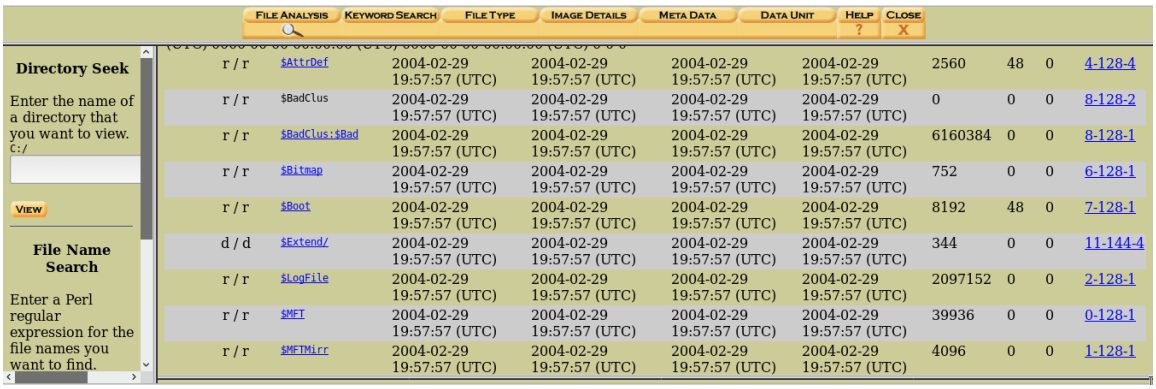
Click on Analyze:



These are some pieces of information about the dump



Now you can analyse the file freely:



Practical Lab 2: Memory Analysis with Volatility

Memory malware analysis is widely used for digital investigation and malware analysis. It refers to the act of analysing a dumped memory image from a targeted machine after executing the malware to obtain multiple numbers of artefacts including network information, running processes, API hooks, kernel loaded modules, Bash history, etc. ... This phase is very important because it is always a good idea to have a clearer understanding of malware capabilities.

- Process list and the associated threads
- Networking information and interfaces (TCP/UDP)
- Kernel modules including the hidden modules
- Opened files in the kernel
- Bash and commands history
- System Calls
- Kernel hooks

To analyse memory You can simply use volatility framework, which is an open-source memory forensics tool written in Python. It is available under GPL. Volatility comes with various plugins and a number of profiles to ease obtaining basic forensic information about memory image files. To download it you can visit this website: The Volatility Foundation - Open Source Memory Forensics or GitHub - volatilityfoundation/volatility

As a hands-on practice, we are going to analyse a memory dump from an infected computer with Volatility. You can find many samples here: https://github.com/volatilityfoundation/volatility/wiki/Memory-Samples

For the demonstration, we are going to analyse a memory dump called " **cridex.vmem**"

```
wget http://files.sempersecurus.org/dumps/cridex_memdump.zip
```

Get info about the memory dump:

```
python vol.py -f cridex.vmem imageinfo
```

```
Volatility Foundation Volatility Framework 2.6.1
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/home/azureuser/volatility/volatility/
cridex.vmem)
                      PAE type : PAE
                           DTB : 0x2fe000L
                          KDBG : 0x80545ae0L
          Number of Processors : 1
     Image Type (Service Pack) : 3
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2012-07-22 02:45:08 UTC+0000
     Image local date and time : 2012-07-21 22:45:08 -0400
```

Get Processes

```
python vol.py -f cridex.vmem psxview
```

```
Volatility Foundation Volatility Framework 2.6.1
Offset(P)  Name                    PID pslist psscan thrdproc pspcid csrss session deskthrd ExitTime
---------- -------------------- ------ ------ ------ -------- ------ ----- ------- -------- --------
0x02498700 winlogon.exe            608 True   True   True     True   True  True    True
0x02511360 svchost.exe             824 True   True   True     True   True  True    True
0x022e8da0 alg.exe                 788 True   True   True     True   True  True    True
0x020b17b8 spoolsv.exe            1512 True   True   True     True   True  True    True
0x0202ab28 services.exe            652 True   True   True     True   True  True    True
0x02495650 svchost.exe            1220 True   True   True     True   True  True    True
0x0207bda0 reader_sl.exe          1640 True   True   True     True   True  True    True
0x025001d0 svchost.exe            1004 True   True   True     True   True  True    True
0x02029ab8 svchost.exe             908 True   True   True     True   True  True    True
0x023fcda0 wuauclt.exe            1136 True   True   True     True   True  True    True
0x0225bda0 wuauclt.exe            1588 True   True   True     True   True  True    True
0x0202a3b8 lsass.exe               664 True   True   True     True   True  True    True
0x023dea70 explorer.exe           1484 True   True   True     True   True  True    True
0x023dfda0 svchost.exe            1056 True   True   True     True   True  True    True
0x024f1020 smss.exe                368 True   True   True     True   False False   False
0x025c89c8 System                    4 True   True   True     True   False False   False
0x024a0598 csrss.exe               584 True   True   True     True   False True    True
```

Processes as Parent/Child

```
sudo python vol.py -f cridex.vmem pstree
```

```
Volatility Foundation Volatility Framework 2.6.1
Offset(P)          Name                PID   PPID PDB        Time created                   Time exited
------------------ ---------------- ------ ------ ---------- ------------------------------ ----------
-------------
0x0000000002029ab8 svchost.exe         908    652 0x079400e0 2012-07-22 02:42:33 UTC+0000

0x000000000202a3b8 lsass.exe           664    608 0x079400a0 2012-07-22 02:42:32 UTC+0000

0x000000000202ab28 services.exe        652    608 0x07940080 2012-07-22 02:42:32 UTC+0000

0x0000000000207bda0 reader_sl.exe     1640   1484 0x079401e0 2012-07-22 02:42:36 UTC+0000

0x00000000020b17b8 spoolsv.exe        1512    652 0x079401c0 2012-07-22 02:42:36 UTC+0000

0x000000000225bda0 wuauclt.exe        1588   1004 0x07940200 2012-07-22 02:44:01 UTC+0000

0x00000000022e8da0 alg.exe             788    652 0x07940140 2012-07-22 02:43:01 UTC+0000

0x00000000023dea70 explorer.exe       1484   1464 0x079401a0 2012-07-22 02:42:36 UTC+0000

0x00000000023dfda0 svchost.exe        1056    652 0x07940120 2012-07-22 02:42:33 UTC+0000
```

Get hidden and terminated Processes

```
sudo python vol.py -f cridex.vmem psscan
```

```
Volatility Foundation Volatility Framework 2.6.1
Offset(P)          Name                PID   PPID PDB        Time created                   Time exited
------------------ ---------------- ------ ------ ---------- ------------------------------ ------------------------------
0x0000000002029ab8 svchost.exe         908    652 0x079400e0 2012-07-22 02:42:33 UTC+0000

0x000000000202a3b8 lsass.exe           664    608 0x079400a0 2012-07-22 02:42:32 UTC+0000

0x000000000202ab28 services.exe        652    608 0x07940080 2012-07-22 02:42:32 UTC+0000

0x000000000207bda0 reader_sl.exe      1640   1484 0x079401e0 2012-07-22 02:42:36 UTC+0000

0x00000000020b17b8 spoolsv.exe        1512    652 0x079401c0 2012-07-22 02:42:36 UTC+0000

0x000000000225bda0 wuauclt.exe        1588   1004 0x07940200 2012-07-22 02:44:01 UTC+0000

0x00000000022e8da0 alg.exe             788    652 0x07940140 2012-07-22 02:43:01 UTC+0000

0x00000000023dea70 explorer.exe       1484   1464 0x079401a0 2012-07-22 02:42:36 UTC+0000

0x00000000023dfda0 svchost.exe        1056    652 0x07940120 2012-07-22 02:42:33 UTC+0000
```

Get DLLs

`sudo python vol.py -f cridex.vmem dlllist`

```
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
System pid:      4
Unable to read PEB for task.
************************************************************************
smss.exe pid:    368
Command line : \SystemRoot\System32\smss.exe


Base             Size  LoadCount LoadTime                       Path
---------- ---------- ---------- ------------------------------ ----
0x48580000     0xf000     0xffff                                \SystemRoot\System32\smss.exe
0x7c900000    0xaf000     0xffff                                C:\WINDOWS\system32\ntdll.dll
************************************************************************
csrss.exe pid:    584
Command line : C:\WINDOWS\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,3072,512 Windows=On
SubSystemType=Windows ServerDll=basesrv,1 ServerDll=winsrv:UserServerDllInitialization,3 ServerDll=winsrv:Con
ServerDllInitialization,2 ProfileControl=Off MaxRequestThreads=16
Service Pack 3

Base             Size  LoadCount LoadTime                       Path
---------- ---------- ---------- ------------------------------ ----
0x4a680000     0x5000     0xffff                                \??\C:\WINDOWS\system32\csrss.exe
0x7c900000    0xaf000     0xffff                                C:\WINDOWS\system32\ntdll.dll
```

Get commandline args

sudo python vol.py -f cridex.vmem cmdline

```
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
System pid:      4
************************************************************************
smss.exe pid:    368
Command line : \SystemRoot\System32\smss.exe
************************************************************************
csrss.exe pid:    584
Command line : C:\WINDOWS\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,3072,512 Windows=On
SubSystemType=Windows ServerDll=basesrv,1 ServerDll=winsrv:UserServerDllInitialization,3 ServerDll=winsrv:Con
ServerDllInitialization,2 ProfileControl=Off MaxRequestThreads=16
************************************************************************
winlogon.exe pid:    608
Command line : winlogon.exe
************************************************************************
services.exe pid:    652
Command line : C:\WINDOWS\system32\services.exe
************************************************************************
lsass.exe pid:    664
Command line : C:\WINDOWS\system32\lsass.exe
************************************************************************
svchost.exe pid:    824
Command line : C:\WINDOWS\system32\svchost -k DcomLaunch
************************************************************************
```

Get SIDs:

```
sudo python vol.py -f cridex.vmem getsids
```

```
System (4): S-1-5-18 (Local System)
System (4): S-1-5-32-544 (Administrators)
System (4): S-1-1-0 (Everyone)
System (4): S-1-5-11 (Authenticated Users)
smss.exe (368): S-1-5-18 (Local System)
smss.exe (368): S-1-5-32-544 (Administrators)
smss.exe (368): S-1-1-0 (Everyone)
smss.exe (368): S-1-5-11 (Authenticated Users)
csrss.exe (584): S-1-5-18 (Local System)
csrss.exe (584): S-1-5-32-544 (Administrators)
csrss.exe (584): S-1-1-0 (Everyone)
csrss.exe (584): S-1-5-11 (Authenticated Users)
winlogon.exe (608): S-1-5-18 (Local System)
winlogon.exe (608): S-1-5-32-544 (Administrators)
winlogon.exe (608): S-1-1-0 (Everyone)
winlogon.exe (608): S-1-5-11 (Authenticated Users)
services.exe (652): S-1-5-18 (Local System)
services.exe (652): S-1-5-32-544 (Administrators)
services.exe (652): S-1-1-0 (Everyone)
services.exe (652): S-1-5-11 (Authenticated Users)
lsass.exe (664): S-1-5-18 (Local System)
lsass.exe (664): S-1-5-32-544 (Administrators)
lsass.exe (664): S-1-1-0 (Everyone)
lsass.exe (664): S-1-5-11 (Authenticated Users)
```

Networking information:

```
sudo python vol.py -f cridex.vmem connscan
```

```
Offset(P)  Local Address             Remote Address            Pid
---------- ------------------------- ------------------------- ---
0x02087620 172.16.112.128:1038       41.168.5.140:8080         1484
0x023a8008 172.16.112.128:1037       125.19.103.198:8080       1484
```

Kernel modules:

```
sudo python vol.py -f cridex.vmem modules
```

```
Offset(V)  Name                Base             Size File
---------- ------------------- ---------- ---------- ----
0x823fc3b0 ntoskrnl.exe        0x804d7000   0x1f8580 \WINDOWS\system32\ntkrnlpa.exe
0x823fc348 hal.dll             0x806d0000    0x20300 \WINDOWS\system32\hal.dll
0x823fc2e0 kdcom.dll           0xf8b9a000     0x2000 \WINDOWS\system32\KDCOM.DLL
0x823fc270 BOOTVID.dll         0xf8aaa000     0x3000 \WINDOWS\system32\BOOTVID.dll
0x823fc208 ACPI.sys            0xf856b000    0x2e000 ACPI.sys
0x823fc198 WMILIB.SYS          0xf8b9c000     0x2000 \WINDOWS\system32\DRIVERS\WMILIB.SYS
0x823fc130 pci.sys             0xf855a000    0x11000 pci.sys
0x823fc0c0 isapnp.sys          0xf869a000     0xa000 isapnp.sys
0x823fc050 compbatt.sys        0xf8aae000     0x3000 compbatt.sys
0x823ed008 BATTC.SYS           0xf8ab2000     0x4000 \WINDOWS\system32\DRIVERS\BATTC.SYS
0x823edf98 intelide.sys        0xf8b9e000     0x2000 intelide.sys
0x823edf28 PCIIDEX.SYS         0xf891a000     0x7000 \WINDOWS\system32\DRIVERS\PCIIDEX.SYS
0x823edeb8 MountMgr.sys        0xf86aa000     0xb000 MountMgr.sys
0x823ede48 ftdisk.sys          0xf853b000    0x1f000 ftdisk.sys
0x823eddd8 dmload.sys          0xf8ba0000     0x2000 dmload.sys
0x823edd70 dmio.sys            0xf8515000    0x26000 dmio.sys
0x823edd00 PartMgr.sys         0xf8922000     0x5000 PartMgr.sys
0x823edc90 VolSnap.sys         0xf86ba000     0xd000 VolSnap.sys
0x823edc28 atapi.sys           0xf84fd000    0x18000 atapi.sys
0x823edbc0 disk.sys            0xf86ca000     0x9000 disk.sys
0x823edb50 CLASSPNP.SYS        0xf86da000     0xd000 \WINDOWS\system32\DRIVERS\CLASSPNP.SYS
0x823edae0 fltMgr.sys          0xf84dd000    0x20000 fltMgr.sys
```

For more information about the most used Volatility commands check these two helpful cheatsheets:

- Volatility foundation CheatSheet_v2.4.pdf
- SANS Volatility-memory-forensics-cheat-sheet.pdf

References:

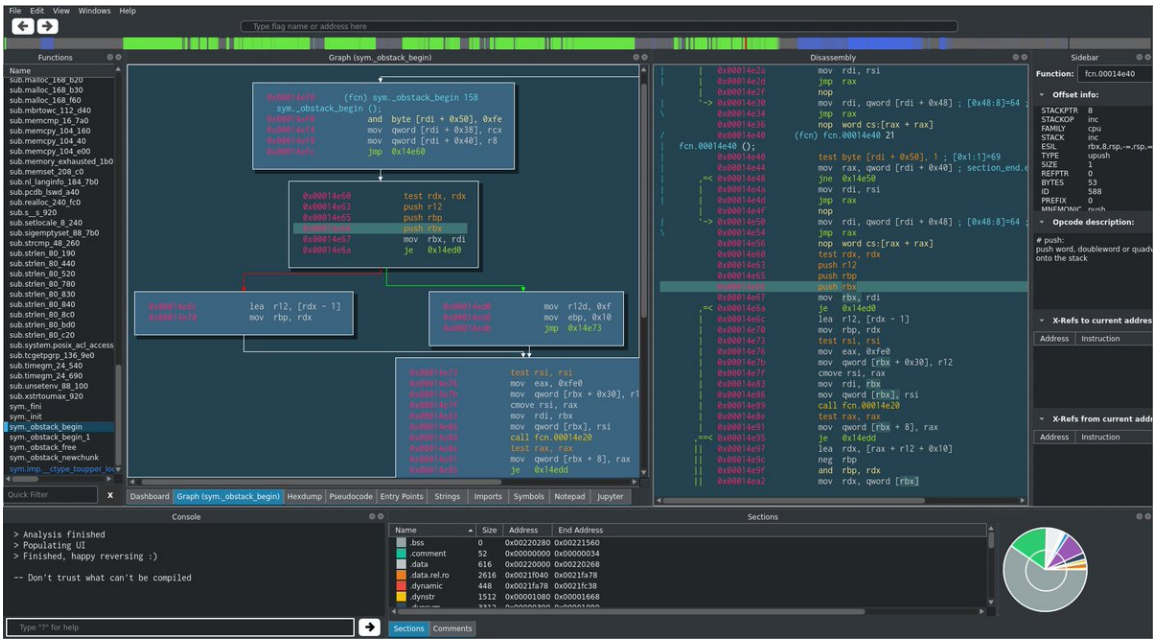
- https://wiki.wireshark.org/SampleCaptures
- Digital Forensics and Incident Response
- Digital Forensics with Kali Linux

Summary

In this module, we discovered what digital forensics is, what are the different steps to perform it, including evidence acquisition and analysis. Later, we explored some well-known digital forensics tools by analyzing some memory dumps using Autopsy and Volatility framework.

How to Perform Static Malware Analysis with Radare2

In this article, we are going to explore how to perform static malware analysis with Radare2.



source

Before diving into technical details let's explore first what is malware analysis and what are the different approaches to perform it.

Malware analysis is the art of determining the functionality, origin and potential impact of a given malware sample, such as a virus, worm, trojan horse, rootkit, or backdoor. As a malware analyst, our main role is to collect all the information about malicious software and have a good understanding of what happened to the infected machines. Like any process, to perform a malware analysis we typically need to follow a certain methodology and a number of steps. To perform Malware Analysis we can go through three phases:

- Static Malware Analysis
- Dynamic Malware Analysis
- Memory Malware Analysis

Static Malware analysis

Static malware analysis refers to the examination of the malware sample without executing it. It consists of providing all the information about the malicious binary. The first steps in the static analysis are knowing the malware size and file type to have a clear vision about the targeted machines, in addition to determining the hashing values, because cryptographic hashes like MD5 or SHA1 can serve as a unique identifier for the sample file. To dive deeper, finding strings, dissecting the binary and reverse-engineering the code of malware using a disassembler like IDA could be a great step to explore how the malware works by studying the program instructions. Malware authors often are trying to make

the work of malware analysts harder so they are always using packers and cryptors to evade detection. That is why, during static analysis, it is necessary to detect them using tools like PEiD.

Dynamic Malware analysis

Performing static analysis is not enough to fully understand malware's true functionality. That is why running the malware in an isolated environment is the next step for the malware analysis process. During this phase, the analyst observes all the behaviours of the malicious binary. Dynamic analysis techniques track all the malware activities, including DNS summary, TCP connections, network activities, syscalls and much more.

Memory Malware analysis

Memory malware analysis is widely used for digital investigation and malware analysis. It refers to the act of analysing a dumped memory image from a targeted machine after executing the malware to obtain multiple numbers of artefacts including network information, running processes, API hooks, kernel loaded modules, Bash history, etc. ... This phase is very important because it is always a good idea to have a clearer understanding of malware capabilities. The first step of memory analysis is memory acquisition by dumping the memory of a machine using a various number of utilities. One of these tools is fmem, which is a kernel module to create a new device called /dev/fmem to allow direct access to the whole memory

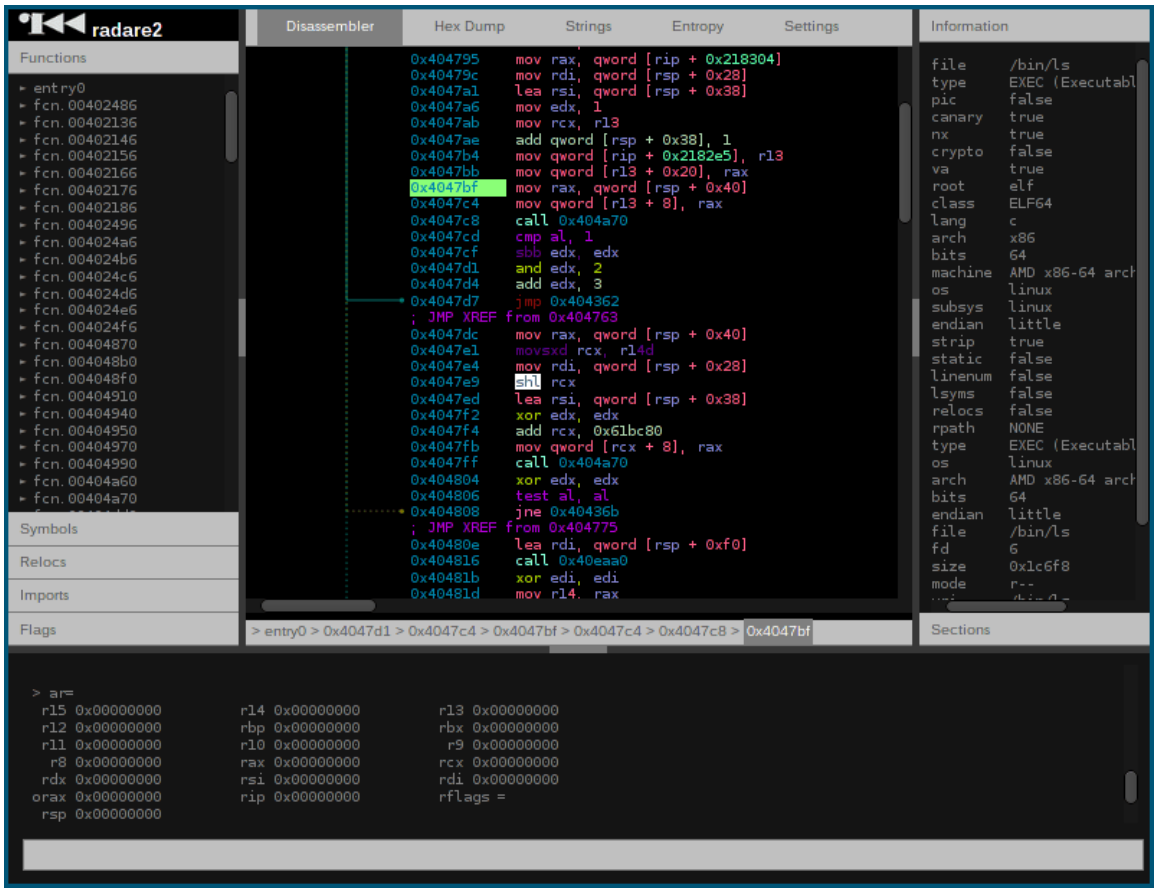
To perform malware analysis you need to build a malware lab. To learn how to do it, I highly recommend you to read my article:

How to perform static malware analysis with Radare2

According to its official

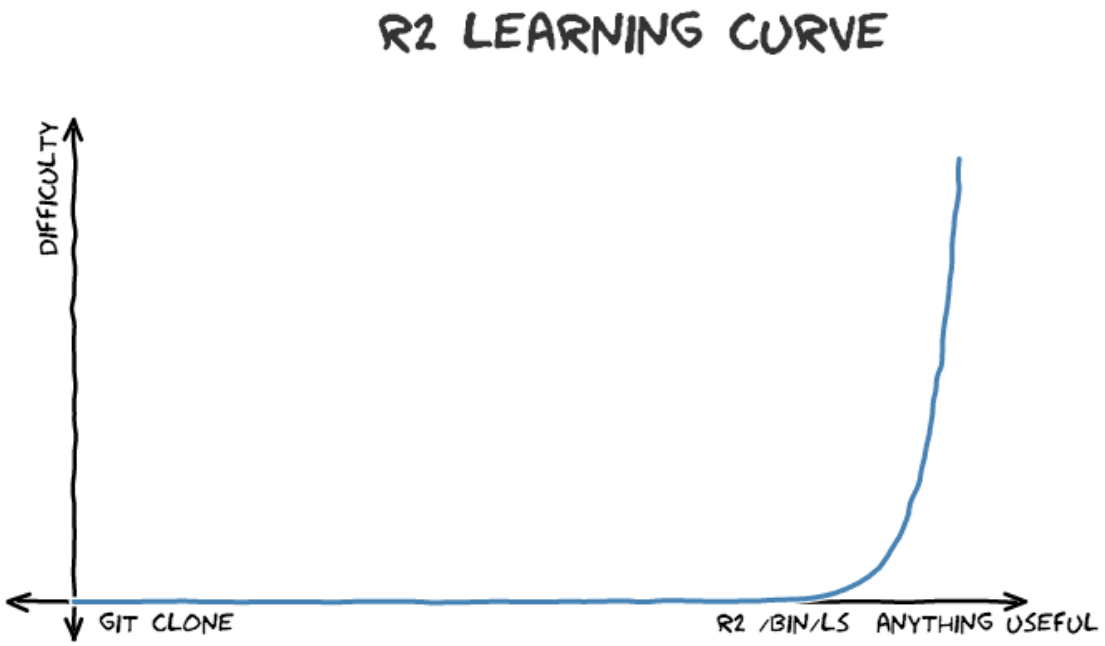
Github account:

Radare2 __ is unix-like __ reverse engineering __ __ framework __ and __ command line__ tools



Source: https://rada.re/r/img/webui.png

It is more than a reverse engineering tool. R2 is able to perform many other tasks. Usually, you will find it hard to learn Radare2 but after a while, you will acquire a good understanding of most of its features.



Source

Let's get started by exploring this great tool. As a demonstration, we are going to learn how to perform some static malware analysis with it. Usually, in the static analysis, we need to perform these tasks and to collect many pieces of information including:

- File type and architecture
- File fingerprinting and hashes
- Strings
- Decoding obfuscation
- Determining Packers and Cryptors
- Header information
- Classification and Yara Rules
- Online AV Scanning (Check the embedded article for more information)

Radare2 installation:

Before using R2 we need to install it first.

$ \<a class="mention" data-id="TMLH8gEnq2rpQcJkH" data-type="Tag" href="/tags/git"\>git clone \<a href="https://github.com/radare/radare2.git" target="_blank" rel="noopener"\>https://github.com/radare/radare2.git\</a\</a

```
cd radare2
```

and install it:

```
$ sys/install.sh
```

Radare2 contains many tools such as **rabin2** , **radiff2** , **rax2** , **rasm2** etc...

If you are using Kali Linux you can use it directly by typing:

```
r2
```

```
Usage: r2 [-ACdfLMnNqStuvwzX] [-P patch] [-p prj] [-a arch] [-b bits] [-i file]
          [-s addr] [-B baddr] [-m maddr] [-c cmd] [-e k=v] file|pid|-|--|=
 --            run radare2 without opening any file
 -             same as 'r2 malloc://512'
 =             read file from stdin (use -i and -c to run cmds)
 -=            perform !=! command to run all commands remotely
 -0            print \x00 after init and every command
 -2            close stderr file descriptor (silent warning messages)
 -a [arch]     set asm.arch
 -A            run 'aaa' command to analyze all referenced code
 -b [bits]     set asm.bits
 -B [baddr]    set base address for PIE binaries
 -c 'cmd..'    execute radare command
 -C            file is host:port (alias for -c+=http://%s/cmd/)
 -d            debug the executable 'file' or running process 'pid'
 -D [backend] enable debug mode (e cfg.debug=true)
 -e k=v        evaluate config var
 -f            block size = file size
 -F [binplug] force to use that rbin plugin
 -h, -hh       show help message, -hh for long
 -H ([var])    display variable
 -i [file]     run script file
 -I [file]     run script file before the file is opened
 -k [OS/kern] set asm.os (linux, macos, w32, netbsd, ...)
```

For the demonstration, I downloaded "Multi-Platform Linux Router DDoS ELF".

As discussed previously first we need to obtain information about the binary:

`rabin2 -I halfnint`

```
arch     x86
binsz    43671
bintype  elf
bits     32
canary   false
class    ELF32
crypto   false
endian   little
havecode true
intrp    /lib/ld-linux.so.2
lang     c
linenum  true
lsyms    true
machine  Intel 80386
maxopsz  16
minopsz  1
nx       true
os       linux
pcalign  0
pic      false
relocs   true
relro    no
rpath    NONE
static   false
stripped false
subsys   linux
va       true
```

To extract the string from the data section type:

`rabin2 -z halfnint`

```
000 0x00006eec 0x0804eeec   4   5 (.rodata) ascii %s%s
001 0x00006ef1 0x0804eef1   7   8 (.rodata) ascii [Intel]
002 0x00006ef9 0x0804eef9  11  12 (.rodata) ascii @UnderWorld
003 0x00006f05 0x0804ef05   5   6 (.rodata) ascii pussy
004 0x00006f0c 0x0804ef0c  45  46 (.rodata) ascii PRIVMSG %s :[login] you are logged in, (%s).\n
005 0x00006f3c 0x0804ef3c  60  61 (.rodata) ascii PRIVMSG %s :[!login] sorry, wrong authenthication password!\n
006 0x00006f79 0x0804ef79  12  13 (.rodata) ascii 146.255.36.1
007 0x00006f88 0x0804ef88  38  39 (.rodata) ascii GET /plain HTTP/1.0\nHost: ipecho.net\n\n
008 0x00006fb2 0x0804efb2  13  14 (.rodata) ascii %d.%d.%*s.%*s
009 0x00006fc8 0x0804efc8  60  61 (.rodata) ascii PRIVMSG %s :[error] one error in your input data, see help!\n
010 0x00007005 0x0804f005  15  16 (.rodata) ascii 178.18.16.96:80
011 0x00007017 0x0804f017  11  12 (.rodata) ascii %d.%d.%d.%d
012 0x00007023 0x0804f023  19  20 (.rodata) ascii /var/run/.liightpid
013 0x00007049 0x0804f049   4   5 (.rodata) ascii ->%s
014 0x00007050 0x0804f050  10  11 (.rodata) ascii eYmUrmyAfG
015 0x0000705b 0x0804f05b   4   5 (.rodata) ascii PASS
016 0x00007060 0x0804f060   6   7 (.rodata) ascii %s %s\n
017 0x00007067 0x0804f067   8   9 (.rodata) ascii NICK %s\n
018 0x00007070 0x0804f070  39  40 (.rodata) ascii USER ass localhost localhost :Stallion\n
019 0x00007098 0x0804f098   9  10 (.rodata) ascii TOPIC %s\n
020 0x000070a8 0x0804f0a8  33  34 (.rodata) ascii %127s%31s%31s%31s%31s%31s%31s%31s
021 0x000070ce 0x0804f0ce   8   9 (.rodata) ascii NICK %s\n
022 0x000070df 0x0804f0df   7   8 (.rodata) ascii :.login
023 0x000070e7 0x0804f0e7  20  21 (.rodata) ascii :.advscan->recursive
024 0x000070fc 0x0804f0fc  20  21 (.rodata) ascii :.advscan->random->b
025 0x00007111 0x0804f111  17  18 (.rodata) ascii :.advscan->random
026 0x00007123 0x0804f123   4   5 (.rodata) ascii PING
027 0x00007128 0x0804f128   7   8 (.rodata) ascii PRIVMSG
028 0x00007130 0x0804f130   8   9 (.rodata) ascii :.logout
```

Load the binary

`radare2 halfnint`

To get information use the " **i** " option. Check all the available gathered information by typing:

`i?`

```
[0x08048ed0]> i?
|Usage: i Get info from opened file (see rabin2's manpage)
| Output mode:
| '*'                 Output in radare commands
| 'j'                 Output in json
| 'q'                 Simple quiet output
| Actions:
| i|ij                Show info of current file (in JSON)
| iA                  List archs
| ia                  Show all info (imports, exports, sections..)
| ib                  Reload the current buffer for setting of the bin (use once only)
| ic                  List classes, methods and fields
| icc                 List classes, methods and fields in Header Format
| iC                  Show signature info (entitlements, ...)
| id[?]               Debug information (source lines)
| idp                 Load pdb file information
| iD lang sym         demangle symbolname for given language
| ie                  Entrypoint
| iee                 Show Entry and Exit (preinit, init and fini)
| iE                  Exports (global symbols)
| iE.                 Current export
| ih                  Headers (alias for iH)
| iHH                 Verbose Headers in raw text
| ii                  Imports
| iI                  Binary info
| ik [query]          Key-value database from RBinObject
| il                  Libraries
| iL [plugin]         List all RBin plugins loaded or plugin details
| im                  Show info about predefined memory allocation
```

For example to collect information about Exports type:

`iE`

```
[0x08048ed0]> iE
[Exports]
059 0x0000cba0 0x08055ba0 GLOBAL    OBJ    4 statfd
060 0x0000cba4 0x08055ba4 GLOBAL    OBJ    8 tm
062 0x0000339e 0x0804b39e GLOBAL   FUNC  333 cmd_exec
066 0x0000cbc0 0x08055bc0 GLOBAL    OBJ   32 psw_y
067 0x00001281 0x08049281 GLOBAL   FUNC   89 wordcmpp
068 0x00008b44 0x08051b44 GLOBAL    OBJ    4 sleeptime
070 0x00001a13 0x08049a13 GLOBAL   FUNC  392 get_spoofed
072 0x00006e20 0x0804ee20 GLOBAL   FUNC    5 __libc_csu_fini
073 0x0000cbe0 0x08055be0 GLOBAL    OBJ 1048560 hosts
074 0x0010cbd0 0x08155bd0 GLOBAL    OBJ    4 max_pids
075 0x0010cbd4 0x08155bd4 GLOBAL    OBJ    4 recv_bytes
076 0x0010cbe0 0x08155be0 GLOBAL    OBJ  128 status_temp
077 0x00001b9b 0x08049b9b GLOBAL   FUNC  230 pidprocess
078 0x00000ed0 0x08048ed0 GLOBAL   FUNC    0 _start
079 0x00002d9a 0x0804ad9a GLOBAL   FUNC 1256 cmd_help
081 0x00001c81 0x08049c81 GLOBAL   FUNC 1081 decode
082 0x0010cc60 0x08155c60 GLOBAL    OBJ    2 founds
084 0x000044cf 0x0804c4cf GLOBAL   FUNC   68 cmd_part
085 0x00004513 0x0804c513 GLOBAL   FUNC  124 cmd_quit
086 0x00008b3c 0x08051b3c GLOBAL    OBJ    4 max_attacks
087 0x0000624f 0x0804e24f GLOBAL   FUNC  198 rand_cmwc
091 0x00006ee0 0x0804eee0 GLOBAL    OBJ    4 _fp_hw
093 0x0010cc64 0x08155c64 GLOBAL    OBJ    4 udpPID
094 0x000016b4 0x080496b4 GLOBAL   FUNC  107 host2ip
095 0x0010cc68 0x08155c68 GLOBAL    OBJ   21 resbuf
099 0x00006ebc 0x0804eebc GLOBAL   FUNC    0 _fini
103 0x0010dfec 0x08156fec GLOBAL    OBJ    4 synTime
105 0x0010cc80 0x08155c80 GLOBAL    OBJ    4 scan_sp
```

Imports:

`ii`

```
[Imports]
   1 0x08048af8  GLOBAL    FUNC __errno_location
   2 0x08048b08  GLOBAL    FUNC sprintf
   3 0x08048b18  GLOBAL    FUNC popen
   4 0x08048b28  GLOBAL    FUNC srand
   5 0x08048b38  GLOBAL    FUNC connect
   6 0x08048b48  GLOBAL    FUNC getpid
   7 0x08048b58  GLOBAL    FUNC pthread_join
   8 0x08048b68  GLOBAL    FUNC pthread_exit
   9 0x08048b78  GLOBAL    FUNC __isoc99_fscanf
  10 0x08048b88  GLOBAL    FUNC signal
  11 0x08048b98    WEAK  NOTYPE __gmon_start__
  12 0x08048000    WEAK  NOTYPE _Jv_RegisterClasses
  13 0x08048ba8  GLOBAL    FUNC __isoc99_sscanf
  14 0x08048bb8  GLOBAL    FUNC strchr
  15 0x08048bc8  GLOBAL    FUNC vsnprintf
  16 0x08048bd8  GLOBAL    FUNC recv
  17 0x08048be8  GLOBAL    FUNC inet_addr
  18 0x08048bf8  GLOBAL    FUNC system
  19 0x08048c08  GLOBAL    FUNC strncpy
  20 0x08048c18  GLOBAL    FUNC sendto
  21 0x08048c28  GLOBAL    FUNC fgets
  22 0x08048c38  GLOBAL    FUNC memset
  23 0x08048c48  GLOBAL    FUNC __libc_start_main
  24 0x08048c58  GLOBAL    FUNC htons
  25 0x08048c68  GLOBAL    FUNC usleep
  26 0x08048c78  GLOBAL    FUNC free
  27 0x08048c88  GLOBAL    FUNC access
  28 0x08048c98  GLOBAL    FUNC fflush
```

Headers:

`ih`

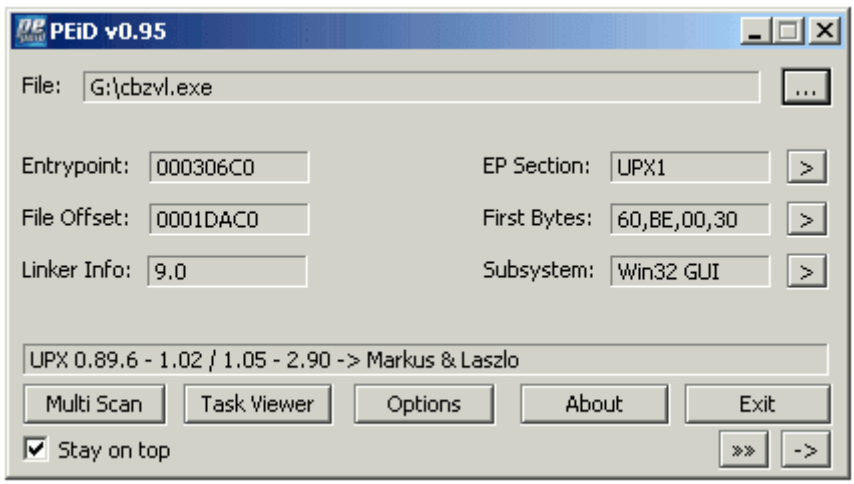
```
[0x08048ed0]> ih
0x00000000  ELF MAGIC   0x464c457f
0x00000010  Type        0x0002
0x00000012  Machine     0x0003
0x00000014  Version     0x00000001
0x00000018  Entrypoint  0x08048ed0
0x0000001c  PhOff       0x00000034
0x00000020  ShOff       0x00008c74
```

To calculate the hashes type:

`rahash2 -a all halfnint`

```
halfnint: 0x00000000-0x0000aa96 md5: ec5556e3026b98aaf0f0a7d53b1a76d6
halfnint: 0x00000000-0x0000aa96 sha1: 5787bc6bf2f77a16109630b8e7055f67b33cb516
halfnint: 0x00000000-0x0000aa96 sha256: 83c01c36ef5dc4c7faf5abb12e295e9ea43393d4567f8d20b3ad176ac53a7bac
halfnint: 0x00000000-0x0000aa96 sha384: 7e829df10e0d8ed509f8cd72c257a9761167877f788e5bb9f8b36fe4dad15d8722a09e18d9bd7480d6f85f489e6ef83f
halfnint: 0x00000000-0x0000aa96 sha512: ff60f6d70568e30768a2d9b3b7969f50aef2667ca3bfd1ce7cda3ca931cc0bd916b28c624ce1bf90a063eff7b3e9fae5d5
ac4744b959e11a07113445ff6a6d5d
halfnint: 0x00000000-0x0000aa96 md4: d87806f5892e2b013ff51f00b1808d37
halfnint: 0x00000000-0x0000aa96 xor: e9
halfnint: 0x00000000-0x0000aa96 xorpair: 6b82
halfnint: 0x00000000-0x0000aa96 parity: 01
halfnint: 0x00000000-0x0000aa96 entropy: 6.08927184
halfnint: 0x00000000-0x0000aa96 hamdist: 01
halfnint: 0x00000000-0x0000aa96 pcprint: 23
halfnint: 0x00000000-0x0000aa96 mod255: d5
halfnint: 0x00000000-0x0000aa96 xxhash: ff07262e
halfnint: 0x00000000-0x0000aa96 adler32: 2cbd3a72
halfnint: 0x00000000-0x0000aa96 luhn: 01
halfnint: 0x00000000-0x0000aa96 crc8smbus: a2
halfnint: 0x00000000-0x0000aa96 crc15can: 4977
halfnint: 0x00000000-0x0000aa96 crc16: 9ae9
halfnint: 0x00000000-0x0000aa96 crc16hdlc: 5508
halfnint: 0x00000000-0x0000aa96 crc16usb: 94d7
halfnint: 0x00000000-0x0000aa96 crc16citt: e715
halfnint: 0x00000000-0x0000aa96 crc24: 537297
halfnint: 0x00000000-0x0000aa96 crc32: 383b75e5
halfnint: 0x00000000-0x0000aa96 crc32c: 5e100313
halfnint: 0x00000000-0x0000aa96 crc32ecma267: 70c8f645
halfnint: 0x00000000-0x0000aa96 crc32bzip2: 5df2f14e
```

To determine the packers usually, we use PEiD



source

But it is a bit outdated, thus, There is Yara support in r2 and PEiD signatures are available in Yara format.

install **libyara**

`r2pm init`

```
r2pm -i yara3-lib
```

Summary

In this module, we explored the different techniques to perform malware analysis. Later we learned how to install an amazing tool called "Radare2" and how to use to perform some static malware analysis tasks.

References:

1. Chiheb Chebbi "Malware Analysis a Machine Learning Approach" eForensics Magazine Issue 07/2017
2. Chiheb Chebbi: How to bypass Machine Learning Malware Detectors with Generative adversarial Networks
3. https://github.com/radare/radare2/blob/master/doc/yara.md

Malware Analysis: How to use Yara rules to detect malware

When performing malware analysis, the analyst needs to collect every piece of information that can be used to identify malicious software. One of the techniques is Yara rules. In this article, we are going to explore Yara rules and how to use them in order to detect malware.

The article outline is the following:

- What is malware analysis
- Static malware analysis techniques
- What is Yara and how to install it
- Detect malware with Yara
- Yara rule structure
- How to write your first Yara rule
- Yara-python

After reading this article you can download this small document that includes other helpful resources: Yara Rules Resources

Malware Analysis

Malware is a complex and malicious piece of software.Its behavior range from basic actions like simple modifications of computer systems to advanced behaviors patterns.

By definition, a malware is a malicious piece of software with the aim of damaging computer systems like data andidentity stealing ,espionage,legitimate users infection and gaining full or limited control to its developer.To have a clear understanding of malware analysis, a malware categorization based on its behavior is a must. Even sometimes we cannot classify a malware because it uses many different functionalities but in general, malware can be divided into many categories some of them are described below:

- **Trojan:** is a malware that appears as a legitimate application
- **Virus:** this type of malware copy itself and infect computer machines
- **Botnets** are networks of compromised machines which are generally controlled by a command and control (C2C) channel
- **Ransomware** this malware encrypts all the data on a computer and ask the victim usually using the cryptocurrency Bitcoin to get the decryption key

- **Spyware** as it is obvious from the name it is a malware that tracks all the user activities including Search history,installed applications
- **Rootkit** enables the attacker to gain an unauthorized access generally administrative to a system.Basically, it is unnoticeable and makes its removal as hard as possible

Malware analysis is the art of determining the functionality, origin and potential impact of a given malware sample, such as a virus, worm, trojan horse, rootkit, or backdoor. As a malware analyst, our main role is to collect all the information about malicious software and have a good understanding of what happened to the infected machines. Like any process, to perform a malware analysis we typically need to follow a certain methodology and a number of steps. To perform Malware Analysis we can go thru three phases:

- Static Malware Analysis
- Dynamic Malware Analysis
- Memory Malware Analysis

Static Malware analysis

Static malware analysis refers to the examination of the malware sample without executing it. It consists of providing all the information about the malicious binary. The first steps in static analysis are knowing the malware size and file type to have a clear vision about the targeted machines, in addition to determining the hashing values, because cryptographic hashes like MD5 or SHA1 can serve as a unique identifier for the sample file. To dive deeper, finding strings, dissecting the binary and reverse engineering the code of malware using a disassembler like IDA could be a great step to explore how the malware works by studying the program instructions. Malware authors often are trying to make the work of malware analysts harder so they are always using packers and cryptors to evade detection. That is why, during static analysis, it is necessary to detect them using tools like PEiD.

In this article, we are going to explore how to use YARA Rules. When performing static malware analysis there are many techniques to classify malware and identify it such as hashes. Another technique is using YARA rules. According to Wikipedia:

" **YARA** is the name of a tool primarily used in malware research and detection. It provides a **rule** -based approach to create descriptions of malware families based on textual or binary patterns. A description is essentially a **Yara rule** name, where these **rules**

Install Yara:

The first step, of course, is installing YARA. If you are using Ubuntu for example, you can simply use

```
sudo apt-get install yara
```

It is already installed on my machine

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
yara is already the newest version (3.7.1-1ubuntu2).
The following packages were automatically installed and are no longer required:
  linux-azure-cloud-tools-4.18.0-1023 linux-azure-cloud-tools-4.18.0-1024
  linux-azure-cloud-tools-5.0.0-1018 linux-azure-headers-4.18.0-1023
  linux-azure-headers-4.18.0-1024 linux-azure-headers-5.0.0-1018
  linux-azure-tools-4.18.0-1023 linux-azure-tools-4.18.0-1024
  linux-azure-tools-5.0.0-1018 linux-headers-5.0.0-1018-azure
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 96 not upgraded.
```

Or you can download the tar file and install it from Github
https://github.com/VirusTotal/yara/releases

```
tar -zxf yara-3.7.1.tar.gz
```

```
cd yara-3.7.1
```

```
./bootstrap.sh ./configure make make install
```

Yara needs the following libraries **automake libtool make** and **gcc** so ensure that you already installed them

```
sudo apt-get install automake libtool make gcc
```

```
azureuser@ubuntu:~$ yara --version
3.7.1
azureuser@ubuntu:~$ yara --help
YARA 3.7.1, the pattern matching swiss army knife.
Usage: yara [OPTION]... [NAMESPACE:]RULES_FILE... FILE | DIR | PID

Mandatory arguments to long options are mandatory for short options too.

  -t,  --tag=TAG                      print only rules tagged as TAG
  -i,  --identifier=IDENTIFIER        print only rules named IDENTIFIER
  -c,  --count                        print only number of matches
  -n,  --negate                       print only not satisfied rules (negate)
  -D,  --print-module-data            print module data
  -g,  --print-tags                   print tags
  -m,  --print-meta                   print metadata
  -s,  --print-strings                print matching strings
  -L,  --print-string-length          print length of matched strings
  -e,  --print-namespace              print rules' namespace
  -p,  --threads=NUMBER               use the specified NUMBER of threads to s
can a directory
  -l,  --max-rules=NUMBER             abort scanning after matching a NUMBER o
f rules
  -d VAR=VALUE                        define external variable
  -x MODULE=FILE                      pass FILE's content as extra data to MOD
```

Let's check if everything went well

Create a dummy rule

```
echo "rule dummy { condition: true }" > my_first_rule
```

```
yara my_first_rule my_first_rule
```

If you get " **dummy my_first_rule**" then everything is Okay!

```
azureuser@ubuntu:~$ echo "rule dummy { condition: true }" > my_first_rule
azureuser@ubuntu:~$ yara my_first_rule my_first_rule
dummy my_first_rule
azureuser@ubuntu:~$
```

The Official YARA documentation can be found here:
https://yara.readthedocs.io/en/stable/gettingstarted.html

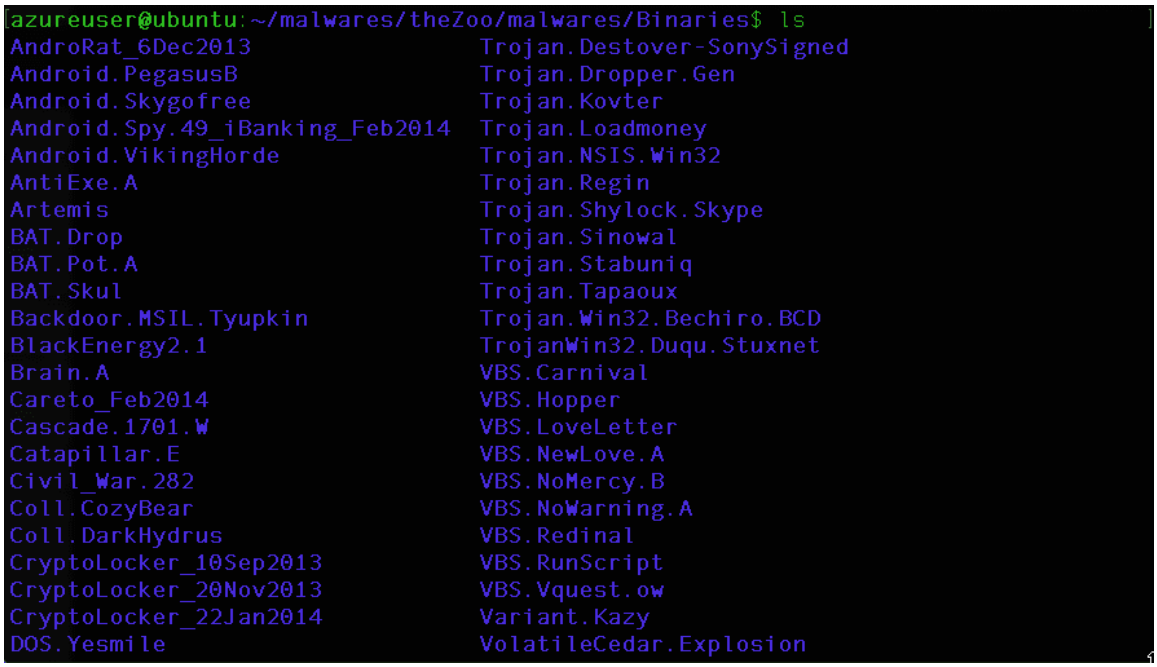
Detect Malware with Yara rules

We already learned that we use Yara rules to detect malware. Let's discover how to do that in a real-world example. For testing purposes, I am going to use malware from a dataset called "theZoo": https://thezoo.morirt.com. The project owners define the repository as follows:

theZoo is a project created to make the possibility of malware analysis open and available to the public. Since we have found out that almost all versions of malware are very hard to come by in a way which will allow analysis, we have decided to gather all of them for you in an accessible and safe way. theZoo was born by Yuval tisf Nativ and is now maintained by Shahak Shalev.

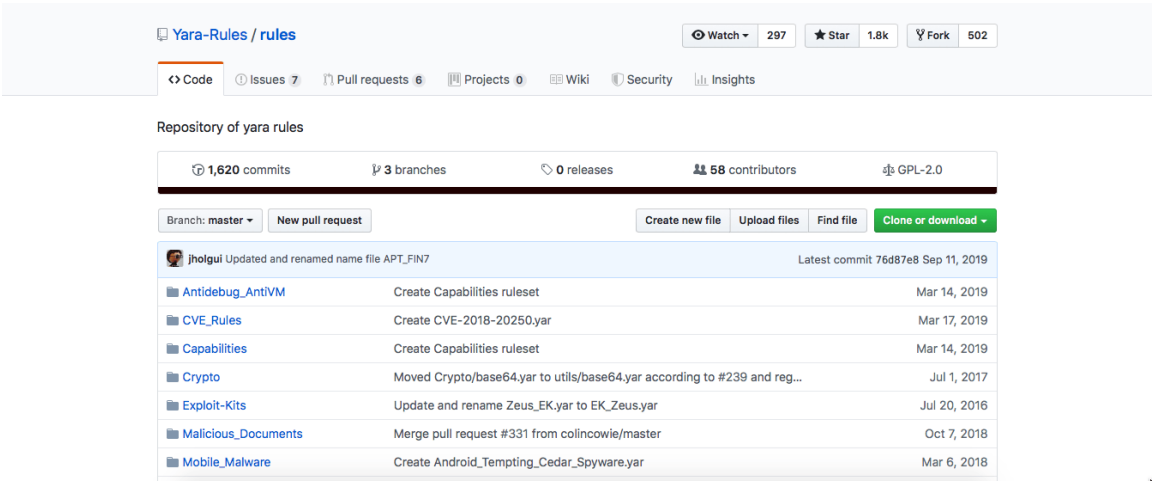
Disclaimer

_ Please remember that these are live and dangerous malware! They come encrypted and locked for a reason! Do NOT run them unless you are absolutely sure of what you are doing! _



Isolation is a security approach provided by many computer systems. It is based on splitting the system into smaller independent pieces to make sure that a compromised sub-system cannot affect the entire entity. Using a sandbox to analyse malware is a wise decision to run untrusted binaries. There are many sandboxes in the wild, such as Cuckoo Sandbox and LIMON, which is an open source sandbox developed by cisco systems Information Security Investigator Monnappa K A as a research project. It is a Python script that automatically collects, analyzes, and reports on Linux malware. It allows one to inspect the Linux malware before execution, during execution, and after execution (post-mortem analysis) by performing static, dynamic and memory analysis using open source tools.

To identify malware we are going to use publically available rules as a demonstration. One of the greatest resources is https://github.com/Yara-Rules/rules



Clone them

```
git clone https://github.com/Yara-Rules/rules
```

```
azureuser@ubuntu:~/rules$ ls
Antidebug_AntiVM            LICENSE                        email
Antidebug_AntiVM_index.yar  Malicious_Documents            email_index.yar
CVE_Rules                   Malicious_Documents_index.yar  index.yar
CVE_Rules_index.yar         Mobile_Malware                 index_gen.sh
Capabilities                Mobile_Malware_index.yar       index_w_mobile.yar
Capabilities_index.yar      Packers                        malware
Crypto                      Packers_index.yar              malware_index.yar
Crypto_index.yar            README.md                      utils
Exploit-Kits                Webshells
Exploit-Kits_index.yar      Webshells_index.yar
```

This project covers the need of a group of IT __Security Researchers__ to have a single repository where different Yara __signatures__ are compiled, classified and kept as up to date as possible, and began as an open source __community__ for collecting Yara rules. Our Yara ruleset is under the GNU-GPLv2 license and open to any user or organization, as long as you use it under this license.

Yara version 3 or higher is required to run the rules.

To detect malware, generally, you need to follow this format

```
yara [OPTIONS] RULES_FILE TARGET
```

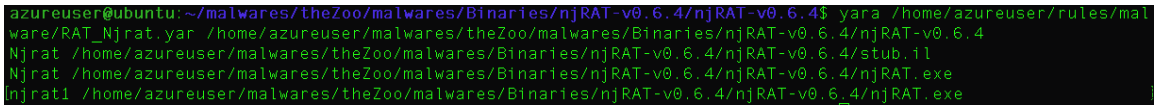
For example to detect NJ-RAT

```
azureuser@ubuntu:~/malwares/theZoo/malwares/Binaries/njRAT-v0.6.4$ ls
njRAT-v0.6.4      njRAT-v0.6.4.pass    njRAT-v0.6.4.zip
njRAT-v0.6.4.md5  njRAT-v0.6.4.sha256
azureuser@ubuntu:~/malwares/theZoo/malwares/Binaries/njRAT-v0.6.4$ cd njRAT-v0.6.4
azureuser@ubuntu:~/malwares/theZoo/malwares/Binaries/njRAT-v0.6.4/njRAT-v0.6.4$ ls
GeoIP.dat       NAudio.dll  Stub.manifest  stub.il
```

Run the following command

```
yara /home/azureuser/rules/malware/RAT\_Njrat.yar
/home/azureuser/malwares/theZoo/malwares/Binaries/njRAT-v0.6.4/njRAT-v0.6.4
```

Yara detect the malicious file



Yara Rules structure

Now let's explore the structure of a Yara rule. Yara rules usually contain:

- Metadata: Information about the rule (Author, development date and so on)
- **Identifiers**
- **Strings identification:** You need to add the strings that YARA needs to look for in order to detect malware.
- **Condition:** this is a logical rule to detect the identified strings and indicators.

For example, this is a skeleton of a simple Yara rule:

```
rule Malware\_Detection
 {
 strings:
 $a = &amp;quot;Sring1&amp;quot;
 $b = &amp;quot;String2&amp;quot;

 condition:
 ($a or $b)
 }
```

You can't use these terms as identifiers:

all, and, any, ascii, at, condition, contains,entrypoint, false, filesize, fullword, for, global, in ,import, include, int8, int16, int32, int8be, int16be,int32be, matches, meta, nocase, not, or, of,private, rule, strings, them, true, uint8, uint16,uint32, uint8be, uint16be, uint32be, wide

This is the Yara rule for the njRAT detection

```
rule Njrat: RAT
{
    meta:
        description = "Njrat"
        author = "botherder https://github.com/botherder"

    strings:
        $string1 = /(F)romBase64String/
        $string2 = /(B)ase64String/
        $string3 = /(C)onnected/ wide ascii
        $string4 = /(R)eceive/
        $string5 = /(S)end/ wide ascii
        $string6 = /(D)ownloadData/ wide ascii
        $string7 = /(D)eleteSubKey/ wide ascii
        $string8 = /(g)et_MachineName/
        $string9 = /(g)et_UserName/
        $string10 = /(g)et_LastWriteTime/
        $string11 = /(G)etVolumeInformation/
        $string12 = /(O)SFullName/ wide ascii
        $string13 = /(n)etsh firewall/ wide
        $string14 = /(c)md\.exe \/k ping 0 & del/ wide
        $string15 = /(c)md\.exe \/c ping 127\.0\.0\.1 & del/ wide
```

```
        $string9 = /(g)et_UserName/
        $string10 = /(g)et_LastWriteTime/
        $string11 = /(G)etVolumeInformation/
        $string12 = /(O)SFullName/ wide ascii
        $string13 = /(n)etsh firewall/ wide
        $string14 = /(c)md\.exe \/k ping 0 & del/ wide
        $string15 = /(c)md\.exe \/c ping 127\.0\.0\.1 & del/ wide
        $string16 = /(c)md\.exe \/c ping 0 -n 2 & del/ wide
        $string17 = {7C 00 27 00 7C 00 27 00 7C}

    condition:
        10 of them
}
```

How to create your first YARA rule

Let's suppose that we are going to create a rule that detects Ardamax Keylogger. First we need to extract the strings using strings command

`strings ArdamaxKeylogger_E33AF9E602CBB7AC3634C2608150DD18`

```
azureuser@ubuntu:~/malwares/theZoo/malwares/Binaries/Keylogger.Ardamax$ ls
ArdamaxKeylogger_E33AF9E602CBB7AC3634C2608150DD18  Keylogger.Ardamax.pass    Keylogger.Ardamax.zip
Keylogger.Ardamax.md5                              Keylogger.Ardamax.sha256
azureuser@ubuntu:~/malwares/theZoo/malwares/Binaries/Keylogger.Ardamax$ strings ArdamaxKeylogger_E33AF9E602CBB
7AC3634C2608150DD18
SrXF
1Rich
.text
`.rdata
@.data
.rsrc
SVWh
SrXFt
X_^[
~$+~8YY
W9^4t
W$YY
G0_^[
W$YY3
F8PWV
        v8j
F0WV
Ht<Ht*H
V(9U
```

Select some strings for demonstration purposes. In my case I am going to select:

- *invalid bit length repeat*
- *??1type_info@@UAE@XZ*
- *.?AVtype_info@@*

Open a text editor and create your rule (FirstRule.yar)

```
rule FirstRule {

meta:
  author = "Chiheb"
  last_updated = "2019"
  category = "Test"
  confidence = "medium"
  description = "This rule was made for a Peerlsyt Article"

strings:
  $a = "invalid bit length repeat" ascii wide nocase
  $b = "??1type_info@@UAE@XZ" ascii wide nocase
  $c = ".?AVtype_info@@" ascii wide nocase

condition:
 ($a or $b or $c)
}
```

wide was added to search for strings encoded with two bytes per character

No case was used to turn off the case-sensitive capability of Yara

Save the rule and run:

```
yara FirstRule.yar ~/malwares/theZoo/malwares/Binaries/Keylogger.Ardamax
```

As you can see Yara detected the malicious file based on our rules:

Yara supports regular expressions thus you can use one of the following expressions

*	**Match 0 or more times**
+	Match 1 or more times
?	Match 0 or 1 time
{n}	Match exactly n times
{n,}	Match at least n times

*	**Match 0 or more times**
{,m}	Match 0 to m times
{n,m}	Match n to m times

Yara Python

It is possible to add Yara capabilities to your python API thanks to a library called "Yara-Python".

With this library you can use YARA from your Python programs. It covers all YARA's features, from compiling, saving and loading rules to __scanning__ files, strings and processes.

To install it:

```
clone https://github.com/VirusTotal/yara-python
```

```
cd yara-python
```

```
python setup.py build
```

```
sudo python  setup.py install
```

This is an example that shows how to include Yara-python in your python application:

```
>>> import yara
>>> rule = yara.compile(source='rule foo: bar {strings: $a = "lmn" condition:
$a}')
>>> matches = rule.match(data='abcdefgjiklmnoprstuvwxyz')
>>> print(matches)
[foo]
>>> print(matches[0].rule)
foo
>>> print(matches[0].tags)
['bar']
>>> print(matches[0].strings)
[(10L, '$a', 'lmn')]
```

Evasion techniques

Black hat Hackers are highly intelligent people. That is why they are looking every day for methods to escape antiviruses and avoid detection.Antiviruses are not totally protection solutions. All the AV vendors are failing to detect advanced persistent attacks no matter how sophisticated their solutions are. Attackers are using many means and tactics to bypass Antivirus protection. Below are some methods used to fool the antiviruses:

- **Obfuscation** is a technique used to make the textual structure of a malware binary hard to read as much as possible. In malware development world is vital to hide what we call the

strings. Strings are significant words usually are URLs, registry keys etc.. To do this, cryptographic standards are used in many cases to achieve this task

- **Binding** is the operation of binding the malware into another legitimate application
- **Crypters and packers** are tools and techniques used to encrypt a malware and keep the antivirus away from peeking inside. Packers some time called executable compression methods are used to make reverse engineering more difficult.

Summary

By now, we explored what is the different malware analysis approaches after a small overview of some types of malicious pieces of software. Later we start exploring Yara rules, their structures, how to detect malware with them and how to create your own first Yara rule. Then we discovered the python interface of Yara. Finally, we learned some AV evasion techniques.

References and further reading:

1. https://www.real0day.com/hacking-tutorials/yara
2. https://0x00sec.org/t/tutorial-creating-yara-signatures-for-malware-detection/5453
3. https://github.com/VirusTotal/yara-python
4. https://seanthegeek.net/257/install-yara-write-yara-rules/
5. https://yara.readthedocs.io/en/v3.4.0/writingrules.html

Getting started with IDA Pro

Reverse engineering is a very important task in information security. It is highly performed in digital forensics, binary exploitation, vulnerability analysis, malware analysis and much more. In this article, we are going to explore an amazing tool called "IDA Pro".

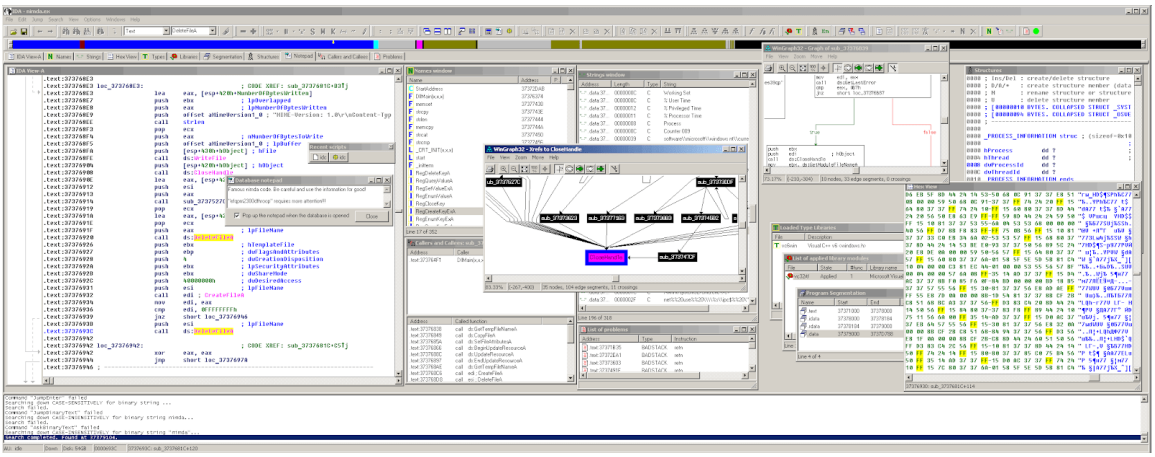
Installation

According to its official website,

'_IDA is a **Windows, Linux** or Mac **OS X** hosted multi-processor **disassembler** and **debugger** that offers so many features it is hard to describe them all' _

There are two versions of IDA:

- Commercial version " **IDA Pro**"
- A free version of it called " **IDA Free**"



source

To install IDA Pro on Windows you just simply need to go to: https://www.hex-rays.com/products/ida/support/download.shtml

IDA Support: Download Center

Evaluation & Freeware versions of IDA

- IDA demo download: evaluate a limited version of our disassembler
- IDA 7.0 Freeware: free for non-commercial use.

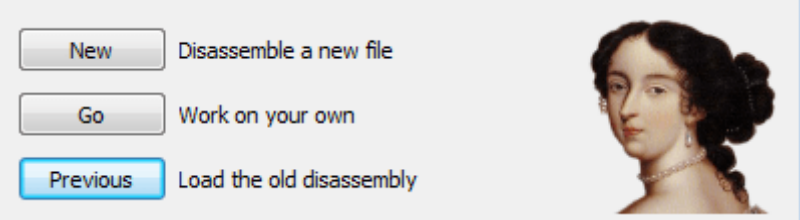
SDK & Utilities

(Some downloads require a password which can be found in the latest IDA download email)

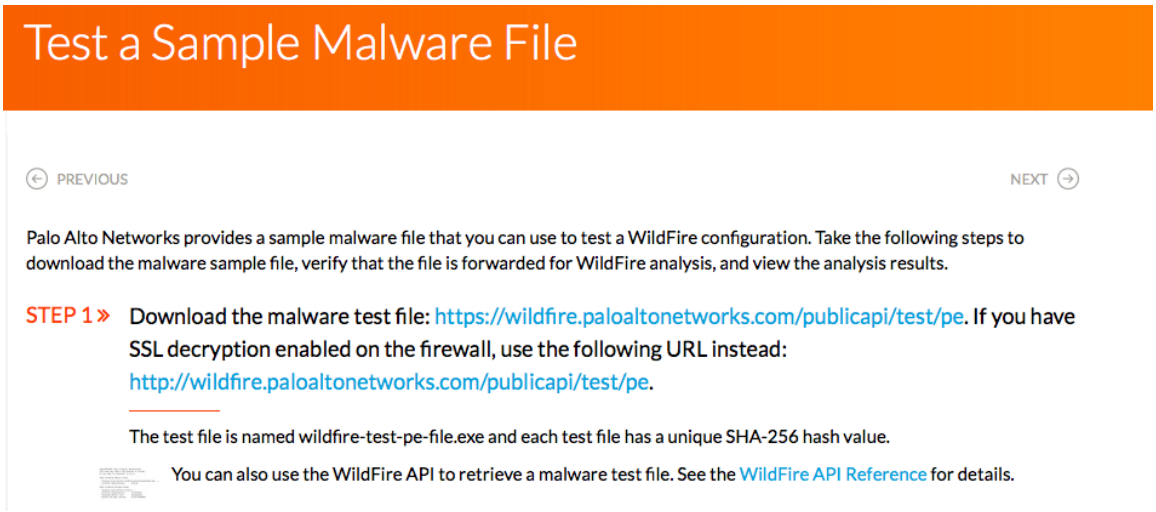
- IDA SDK 7.3: develop processor modules, loaders and extensions - **extended with the source of 30+ modules and 20+ loaders**. Please check out the SDK documentation online (or download the zip file for offline use).
- Flair 7.3 add your own compiler libraries to the FLIRT engine.
- Tilib 7.3 create your own type libraries.
- Loadint 7.3 create your own disassembler comment databases
- idsutils 7.3 create your own IDS files from DLLs.
- ios_deploy iOS helper utility to manipulate iOS devices
- PIN tool: the source code of our PIN tool. It creates a debugger backend out of Intel's PIN framework
- TVision 2015 library for the IDA text interface (source code)
- Qwingraph v1.10: source code the Wingraph we use and modified (GPL).

After installing it you can start it from its desktop shortcut

Once you start it, you will have the choice to work on a new project and load an old disassembly

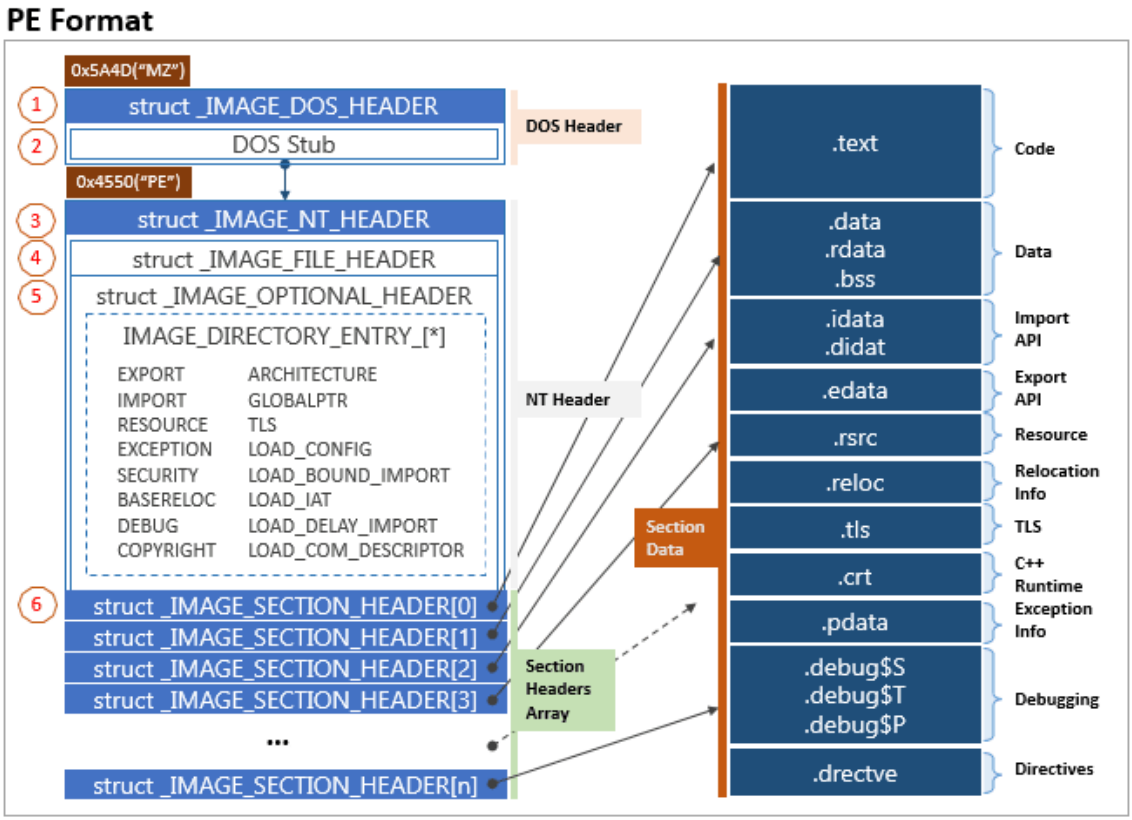


As a demonstration, we are going to disassemble a simple malicious PE file from Paloalto Networks. You can download it from here: https://docs.paloaltonetworks.com/wildfire/7-1/wildfire-admin/submit-files-for-wildfire-analysis/test-a-sample-malware-file



Don't forget to test the file on a sandbox or a VM

Portable Executable (**PE**) files are file formats for executables, DDLs, and object codes used in 32-bit and 64-bit versions of Windows. They contain many useful pieces of information for malware analysts, including imports, exports, time-date stamps, subsystems, sections, and resources. The following is the basic structure of a PE file:



Source: pe_format.png

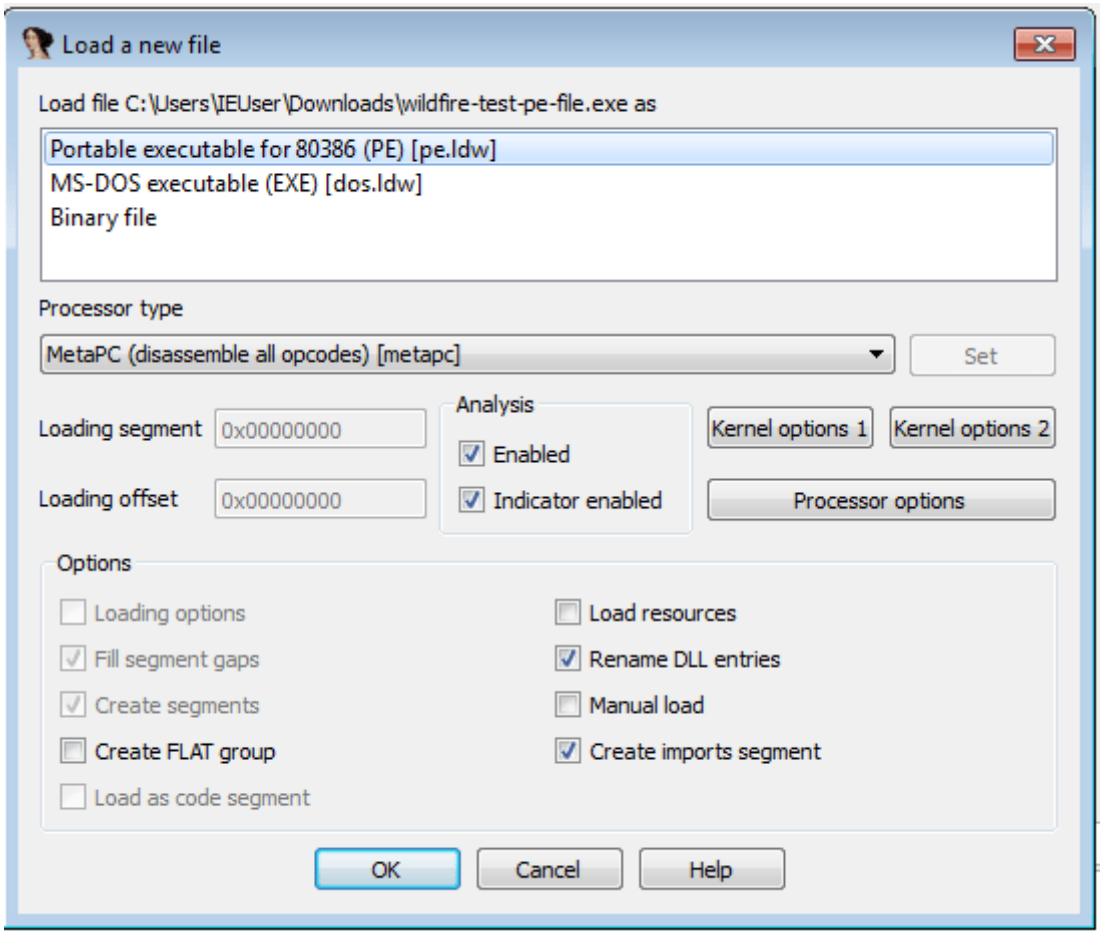
Some of the components of a PE file are as follows:

DOS Header : This starts with the first 64 bytes of every PE file, so DOS can validate the executable and can run it in the DOS stub mode.

PE Header : This contains information, including the location and size of the code.

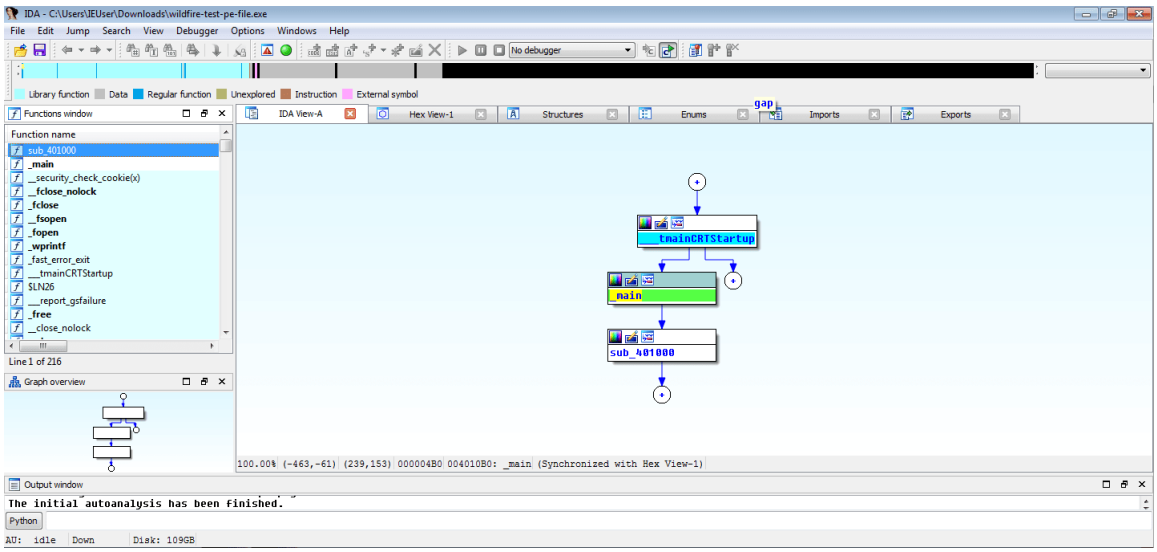
PE Sections They contain the main contents of the file.

Load the PE file:



As you can see from the previous screenshot, IDA Pro is able to detect the file type automatically.

Press "OK" and will be guided to the main interface:



If you load a file, IDA will create a database "*idb*". The database contains:

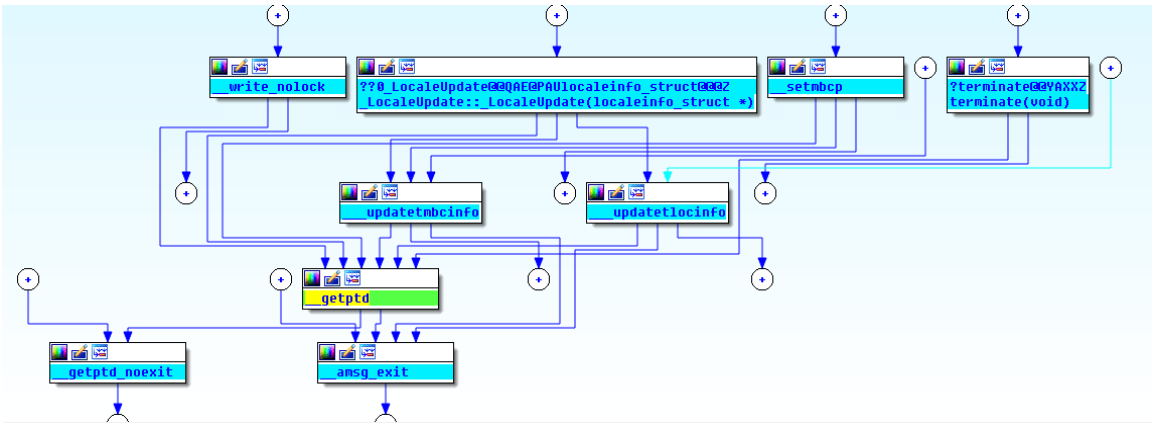
- **Name.id0**

- **name.id1**
- **name.nam**
- **Name.til**

The main interface contains many views and windows:

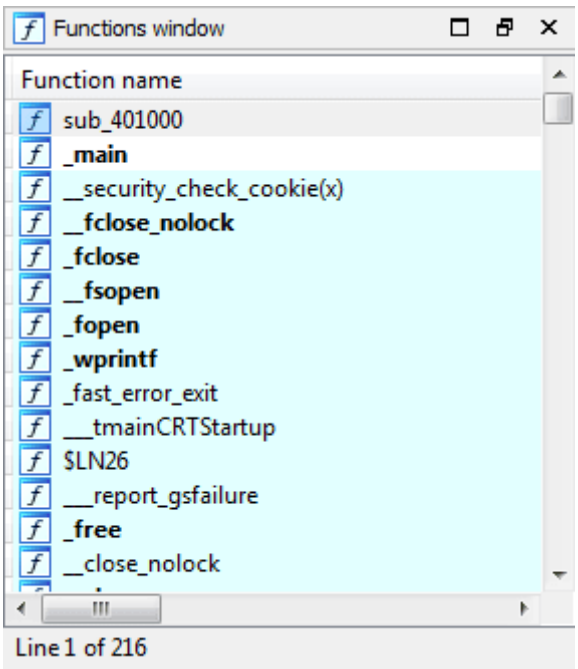
This bar called "the navigation band" illustrates the memory space used by the binary

There is also a graph view to display functions as graphs and sub-graphs



Functions Window:

It lists all the recognizable functions by IDA pro



Imports

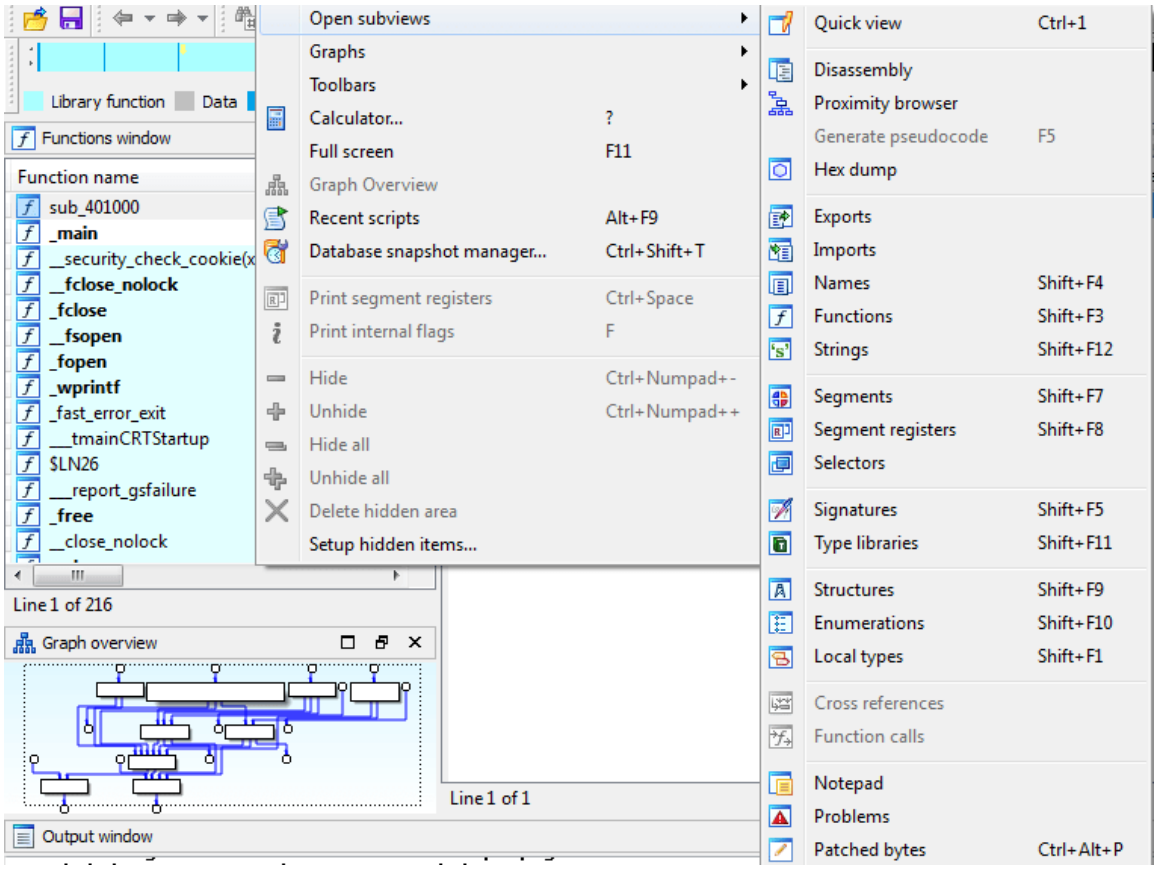
It shows the imported libraries by the loaded binary

Address	Ordinal	Name	Library
0040A000		RegSetValueExW	ADVAPI32
0040A004		RegCloseKey	ADVAPI32
0040A008		RegCreateKeyExW	ADVAPI32
0040A010		GetCommandLineA	KERNEL32
0040A014		HeapSetInformation	KERNEL32
0040A018		TerminateProcess	KERNEL32
0040A01C		GetCurrentProcess	KERNEL32
0040A020		UnhandledExceptionFilter	KERNEL32
0040A024		SetUnhandledExceptionFilter	KERNEL32
0040A028		IsDebuggerPresent	KERNEL32
0040A02C		GetLastError	KERNEL32
0040A030		HeapFree	KERNEL32
0040A034		CloseHandle	KERNEL32
0040A038		EncodePointer	KERNEL32
0040A03C		DecodePointer	KERNEL32
0040A040		EnterCriticalSection	KERNEL32
0040A044		LeaveCriticalSection	KERNEL32
0040A048		InitializeCriticalSectionAndSpinCount	KERNEL32
0040A04C		RtlUnwind	KERNEL32
0040A050		GetProcAddress	KERNEL32
0040A054		GetModuleHandleW	KERNEL32
0040A058		ExitProcess	KERNEL32
0040A05C		WriteFile	KERNEL32

The following is the text view where data is represented as disassembly

```
.text:00408645 ; =============== S U B R O U T I N E =======================================
.text:00408645
.text:00408645 ; Attributes: bp-based frame
.text:00408645
.text:00408645 sub_408645      proc near               ; CODE XREF: __tsopen_nolock+3E↑p
.text:00408645
.text:00408645 arg_0           = dword ptr  8
.text:00408645
.text:00408645                 mov     edi, edi
.text:00408647                 push    ebp
.text:00408648                 mov     ebp, esp
.text:0040864A                 mov     eax, [ebp+arg_0]
.text:0040864D                 test    eax, eax
.text:0040864F                 jnz     short loc_408666
.text:00408651                 call    __errno
.text:00408656                 mov     dword ptr [eax], 16h
.text:0040865C                 call    __invalid_parameter_noinfo
.text:00408661                 push    16h
.text:00408663                 pop     eax
.text:00408664                 pop     ebp
.text:00408665                 retn
.text:00408666 ; ---------------------------------------------------------------------------
.text:00408666
.text:00408666 loc_408666:                             ; CODE XREF: sub_408645+A↑j
.text:00408666                 mov     ecx, dword_40EB3C
.text:0040866C                 mov     [eax], ecx
00007A45 00408645: sub_408645 (Synchronized with Hex View-1)
```

You can find a lot of other available views: **view -\> Open Subviews**

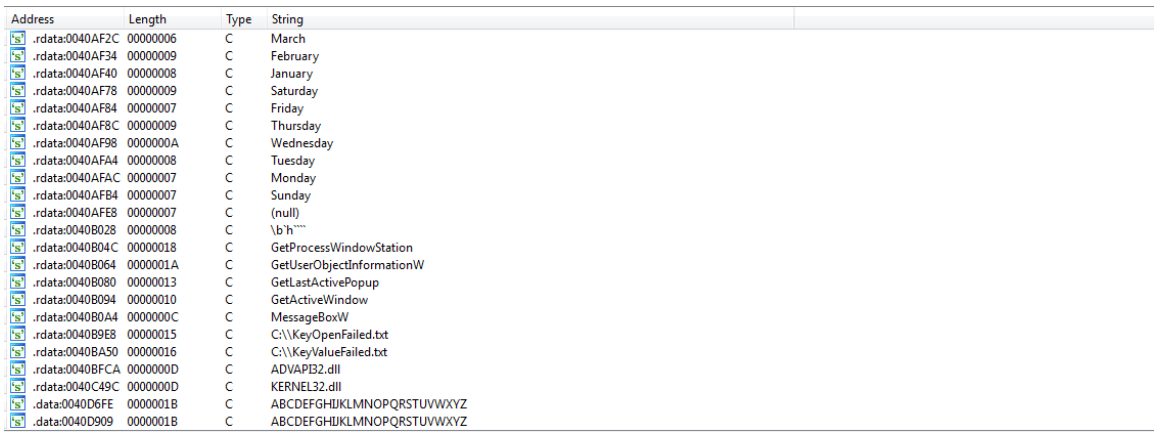


To facilitate the navigation you can simply use the IDA shortcuts including:

Go to a new window: Alt+Enter Text: Alt+T Names: Shift+F4 Functions: Shift+F3

You can find the full list here: Datarescue Interactive Disassembler (IDA) Pro Quick Reference Sheet

Based on its great capabilities IDA Pro is very helpful when it comes to Malware Analysis since it gives you the ability to extract many pieces of information including Strings (F21), imports, exports, graph flows and so on:



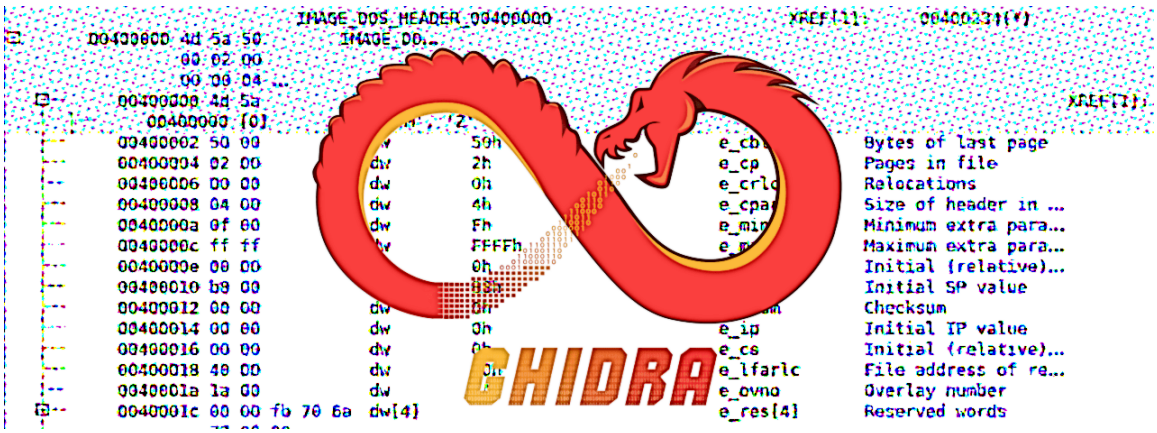
Address	Length	Type	String
.rdata:0040AF2C	00000006	C	March
.rdata:0040AF34	00000009	C	February
.rdata:0040AF40	00000008	C	January
.rdata:0040AF78	00000009	C	Saturday
.rdata:0040AF84	00000007	C	Friday
.rdata:0040AF8C	00000009	C	Thursday
.rdata:0040AF98	0000000A	C	Wednesday
.rdata:0040AFA4	00000008	C	Tuesday
.rdata:0040AFAC	00000007	C	Monday
.rdata:0040AFB4	00000007	C	Sunday
.rdata:0040AFE8	00000007	C	(null)
.rdata:0040B028	00000008	C	\b`h```
.rdata:0040B04C	00000018	C	GetProcessWindowStation
.rdata:0040B064	0000001A	C	GetUserObjectInformationW
.rdata:0040B080	00000013	C	GetLastActivePopup
.rdata:0040B094	00000010	C	GetActiveWindow
.rdata:0040B0A4	0000000C	C	MessageBoxW
.rdata:0040B9E8	00000015	C	C:\\KeyOpenFailed.txt
.rdata:0040BA50	00000016	C	C:\\KeyValueFailed.txt
.rdata:0040BFCA	0000000D	C	ADVAPI32.dll
.rdata:0040C49C	0000000D	C	KERNEL32.dll
.data:0040D6FE	0000001B	C	ABCDEFGHIJKLMNOPQRSTUVWXYZ
.data:0040D909	0000001B	C	ABCDEFGHIJKLMNOPQRSTUVWXYZ

If you want to explore another great tool, I highly recommend you to take a look at my article:" **How to Perform Static Malware Analysis with Radare2**"

In this article, we did a high-level overview of IDA PRO

Getting Started with Reverse Engineering using Ghidra

In this article, we are going to explore how to download Ghidra, install it and use it to perform many important tasks such as reverse engineering, binary analysis and malware analysis.



Source

But first what is Ghidra exactly?

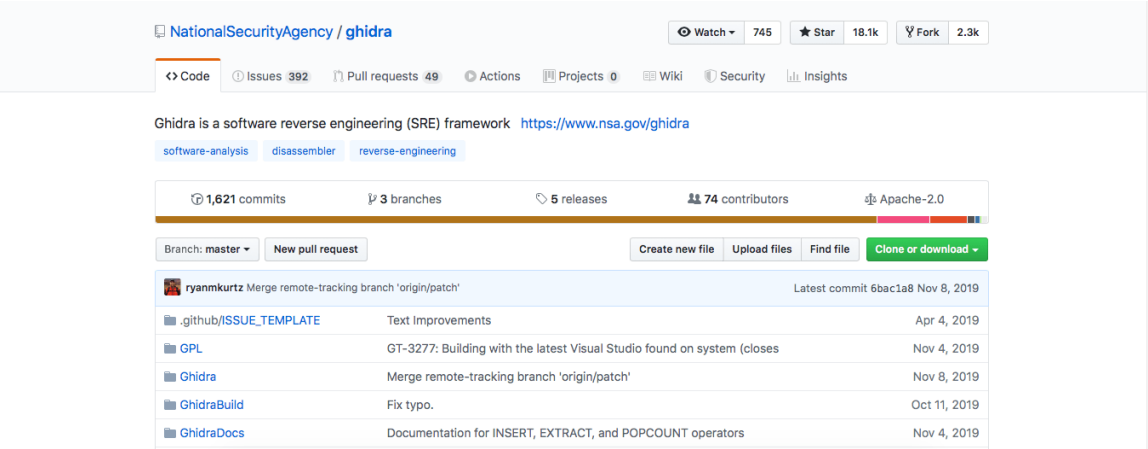
According to its official Github repository:

"Ghidra is a software reverse engineering (SRE) framework created and maintained by the National Security AgencyResearch Directorate. This framework includes a suite of full-featured, high-end software analysis tools that enable users to analyze compiled code on a variety of platforms including Windows, macOS, and Linux. Capabilities include disassembly, assembly, decompilation, graphing, and scripting, along with hundreds of other features. Ghidra supports a wide variety of processor instruction sets and executable formats and can be run in both user-interactive and automated modes. Users may also develop their own Ghidra plug-in components and/or scripts using Java or Python.

In support of NSA's Cyber Security mission, Ghidra was built to solve scaling and teaming problems on complex SRE efforts, and to provide a customizable and extensible SRE research

platform. NSA has applied Ghidra SRE capabilities to a variety of problems that involve analyzing malicious code and generating deep insights for SRE analysts who seek a better understanding of potential vulnerabilities in networks and systems.

https://github.com/NationalSecurityAgency/ghidra



The official website of the project is https://ghidra-sre.org:

As you can notice from the official description that this tool was developed and maintained by the US NSA (National Security Agency) which leads us to think about if this tool is secure. Check this post if you didn't know what i am talking about:

Compilation example with a C Program:

Before diving into the fundamentals of reverse engineering with this powerful tool (Ghidra) , let's explore the compiling phases in order to get an executable and some important terminologies.

Wikipedia defines Reverse engineering as follows:

"_Reverse engineering, also called back engineering, is the process by which a human-made object is deconstructed to reveal its designs, architecture **, or to extract knowledge** __from the object; similar to scientific research, the only difference being that scientific research is about a natural phenomenon." _

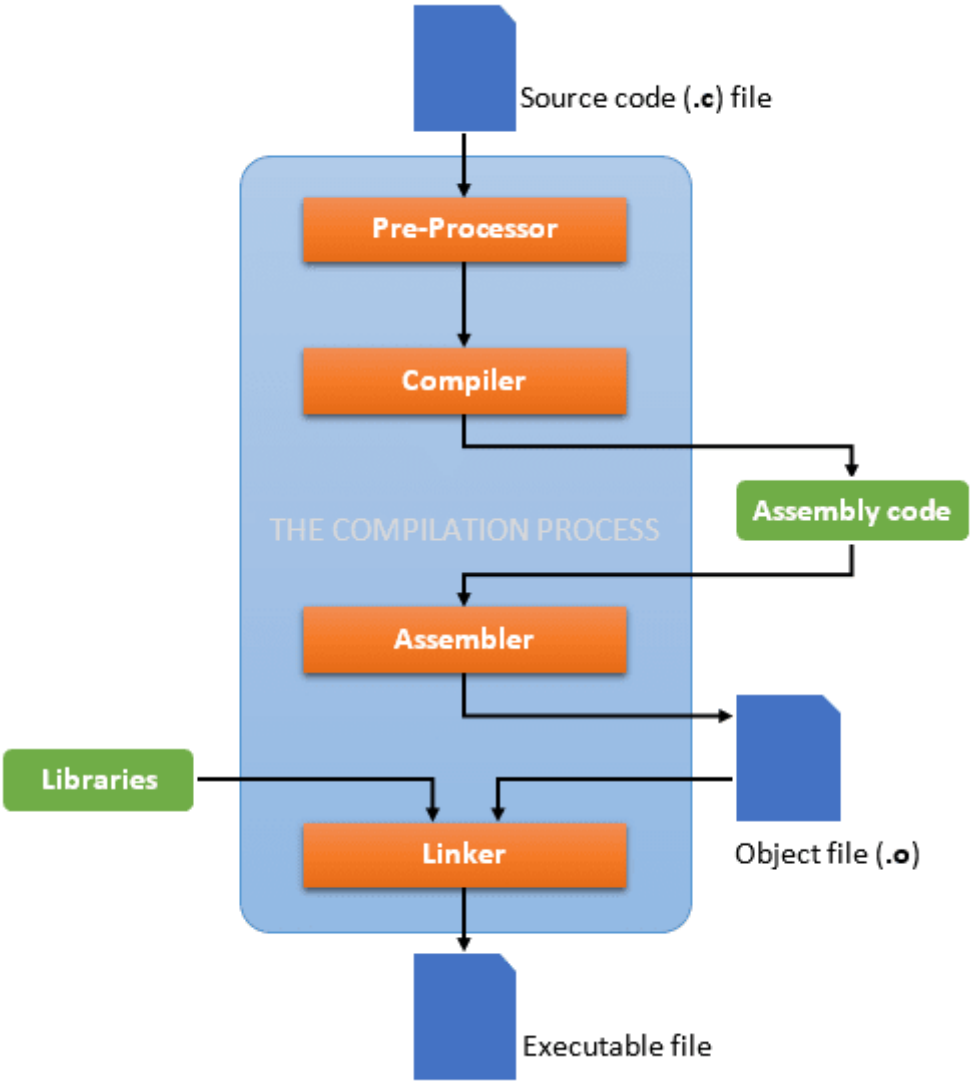
Compilers: convert high-level code to assembly code

Assemblers: convert assembly code to machine code

Linkers: take the object files in order to generate the executable

Disassemblers: convert machine code to assembly code

The phases are represented in the following graph:



Figure

As a demonstration, let's compile a simple c program. The most known easy program is simply a " **hello world!**" program

Create a **hello.c** program:

```
#include <stdio.h>
void main(void)
{
    printf ("hello world!\n");
}
```

Now let's compile it and link it with gcc

```
gcc -o helloWorld hello.c
```

Run the executable

```
./helloWorld
```

```
azureuser@Kali-linux:~$ ./helloWorld
hello world!
```

How to install Ghidra?

To use Ghidra we need to install it of course. As technical requirements, you need the following

Hardware

- 4 GB RAM
- 1 GB storage (for installed Ghidra binaries)
- Dual monitors strongly suggested

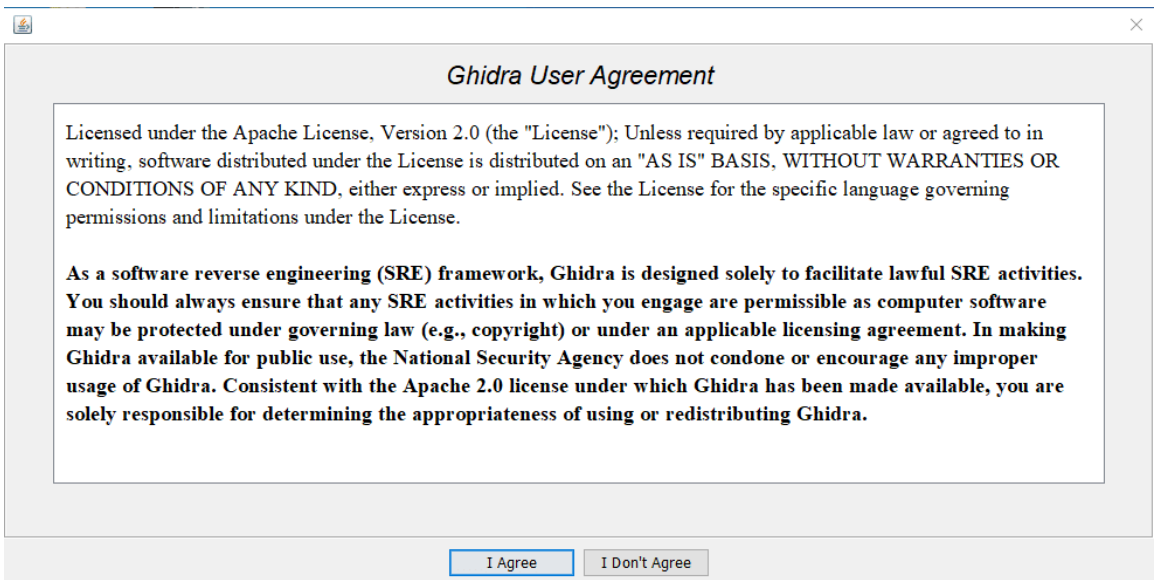
Software

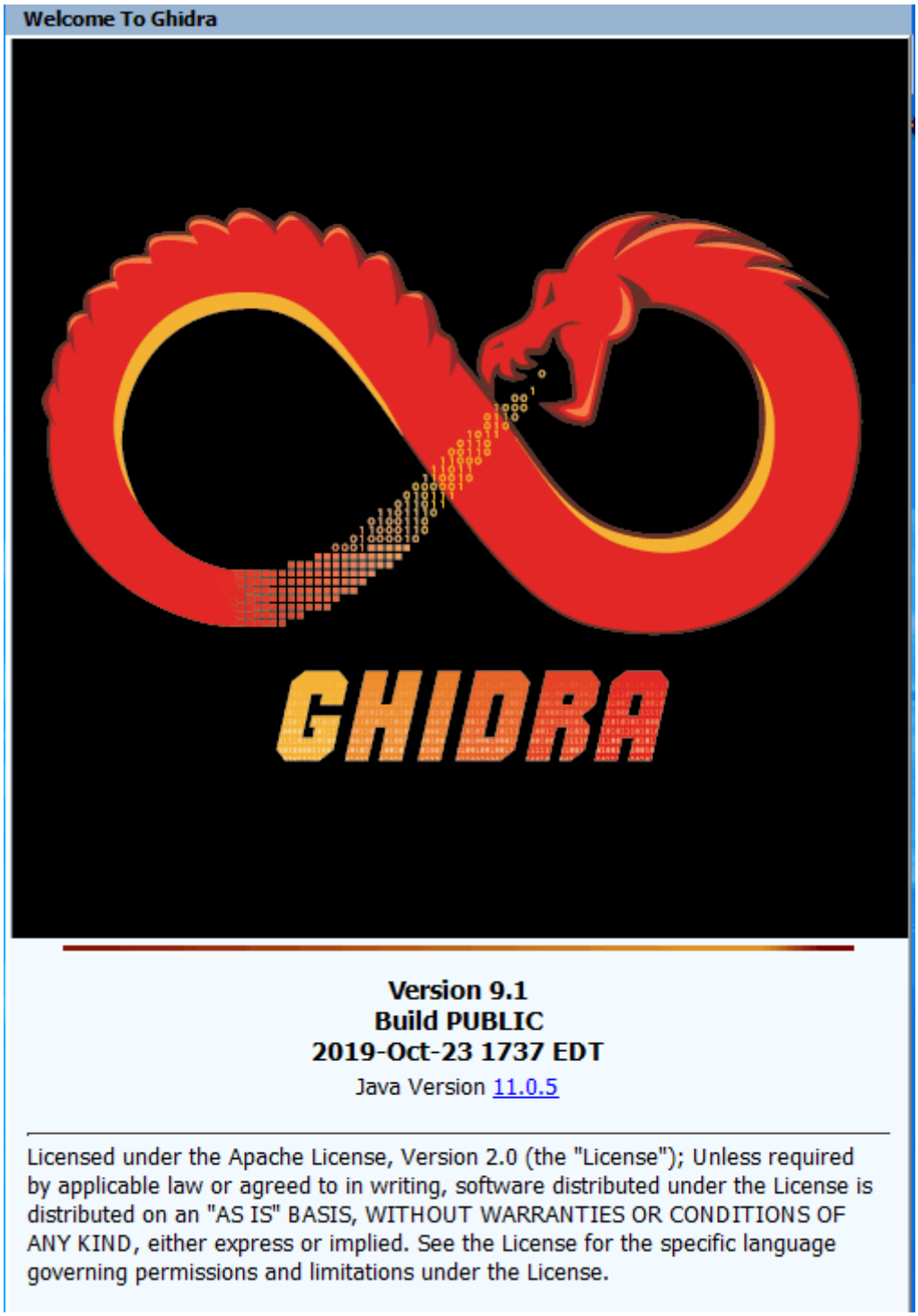
- Java 11 64-bit Runtime and Development Kit (JDK)

Go to Download Ghidra v9.1

Download it and install Java JDK

Go to the installation folder and run the Ghidra bat file





For more information about the installation steps you can check Ghidra official documentation: https://ghidra-sre.org/InstallationGuide.html

Reverse engineering example (CrackMe Challenge):

We learned the compilation phases in order to generate a fully working binary. Now it is time to continue our learning experience with acquiring some fundamentals about reverse engineering. That is why we are going to download a small and easy CrackMe challenge and we will try to

understand what is doing and how it works in order to find the correct password to solve the challenges.

The challenge that we are going to solve is a part of this free and publicly available training materials: https://github.com/Maijin/Workshop2015

We are going to follow Here Be Dragons: Reverse Engineering with Ghidra

Download the GitHub repository, go to /IOLI-crackme/bin-win32 and you will find the challenge binaries.

```
C:\Users\chiheb\Desktop\Workshop2015-master\IOLI-crackme\bin-win32>dir
 Le volume dans le lecteur C n'a pas de nom.
 Le numéro de série du volume est AED9-D96B

 Répertoire de C:\Users\chiheb\Desktop\Workshop2015-master\IOLI-crackme\bin-win32

04/09/2015  21:24    <DIR>          .
04/09/2015  21:24    <DIR>          ..
04/09/2015  21:24            24 440 crackme0x00.exe
04/09/2015  21:24            24 264 crackme0x01.exe
04/09/2015  21:24            24 264 crackme0x02.exe
04/09/2015  21:24            24 318 crackme0x03.exe
04/09/2015  21:24            24 650 crackme0x04.exe
04/09/2015  21:24            24 668 crackme0x05.exe
04/09/2015  21:24            24 863 crackme0x06.exe
04/09/2015  21:24            12 288 crackme0x07.exe
04/09/2015  21:24            25 411 crackme0x08.exe
04/09/2015  21:24            12 288 crackme0x09.exe
              10 fichier(s)          221 454 octets
               2 Rép(s)  313 958 129 664 octets libres
```

We are going to reverse " **Crackme0x01**" file.

Let's open it directly using the command line terminal:

Enter the binaries folder and type:

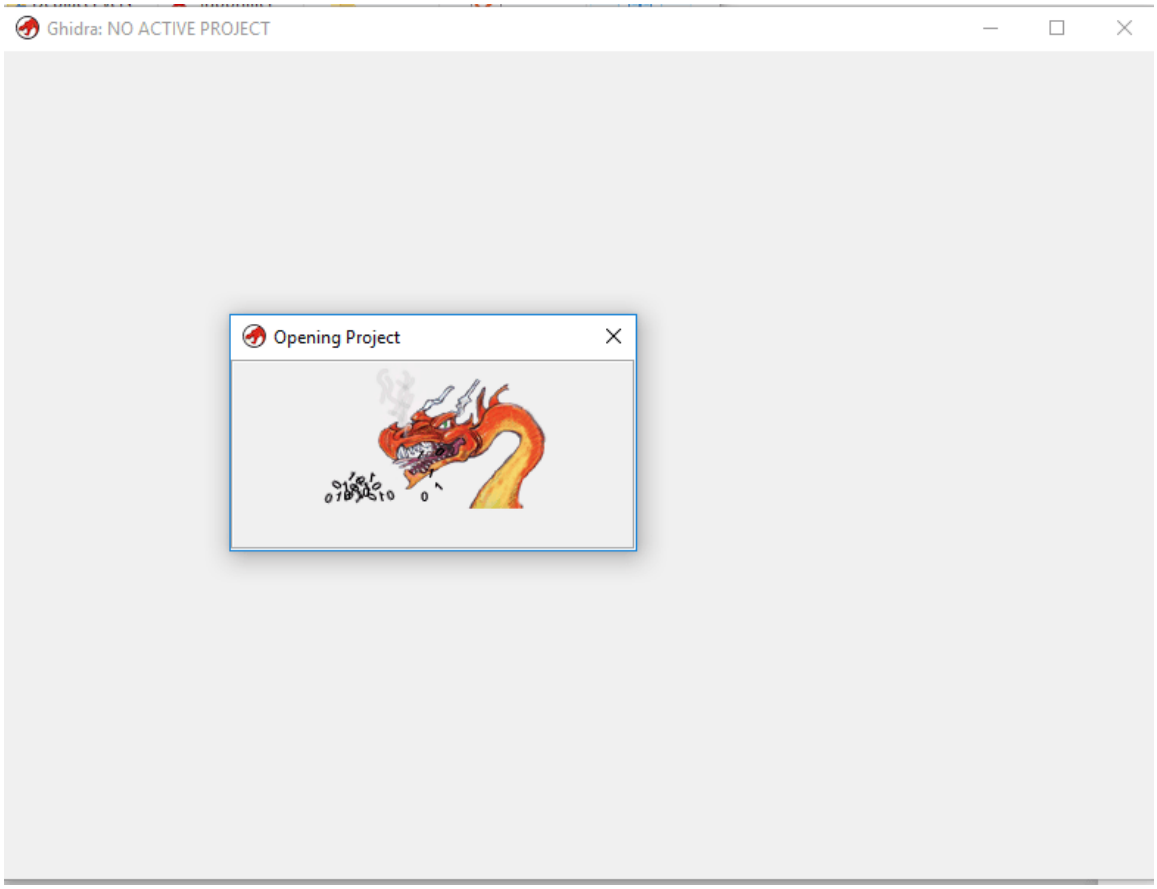
`Crackme0x01.exe`

Enter a random password. In my case I entered "root" but i get an "Invalid Password!" error message

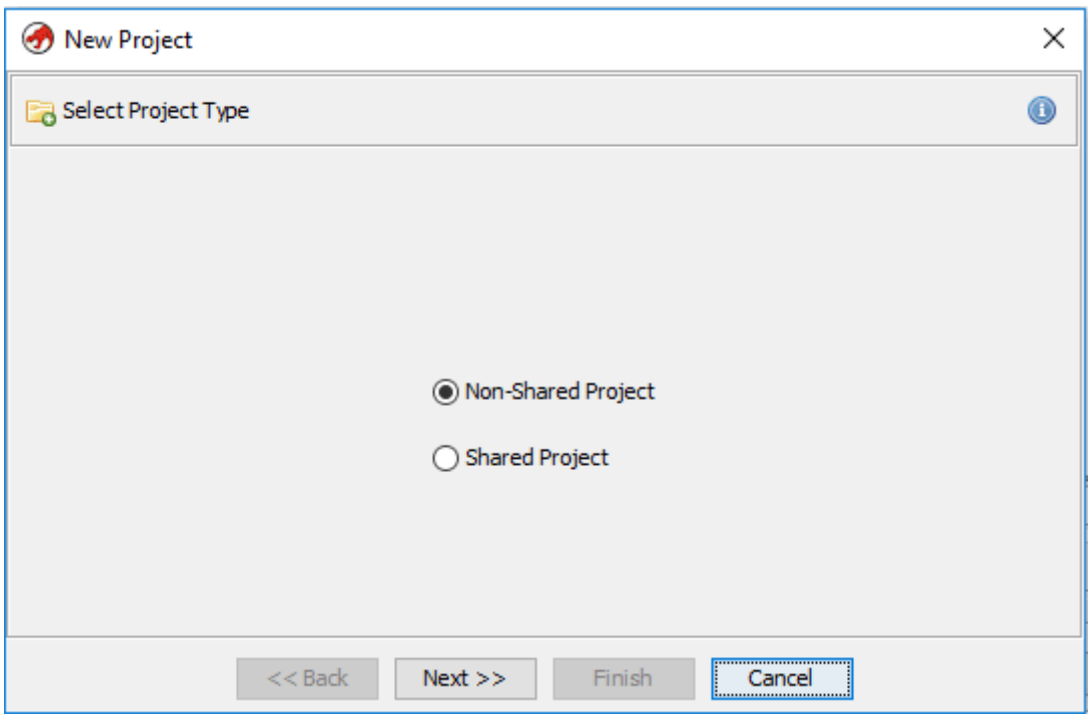
```
C:\Users\chiheb\Desktop\Workshop2015-master\IOLI-crackme\bin-win32>crackme0x01.exe
IOLI Crackme Level 0x01
Password: root
Invalid Password!
```

Then let's crack it

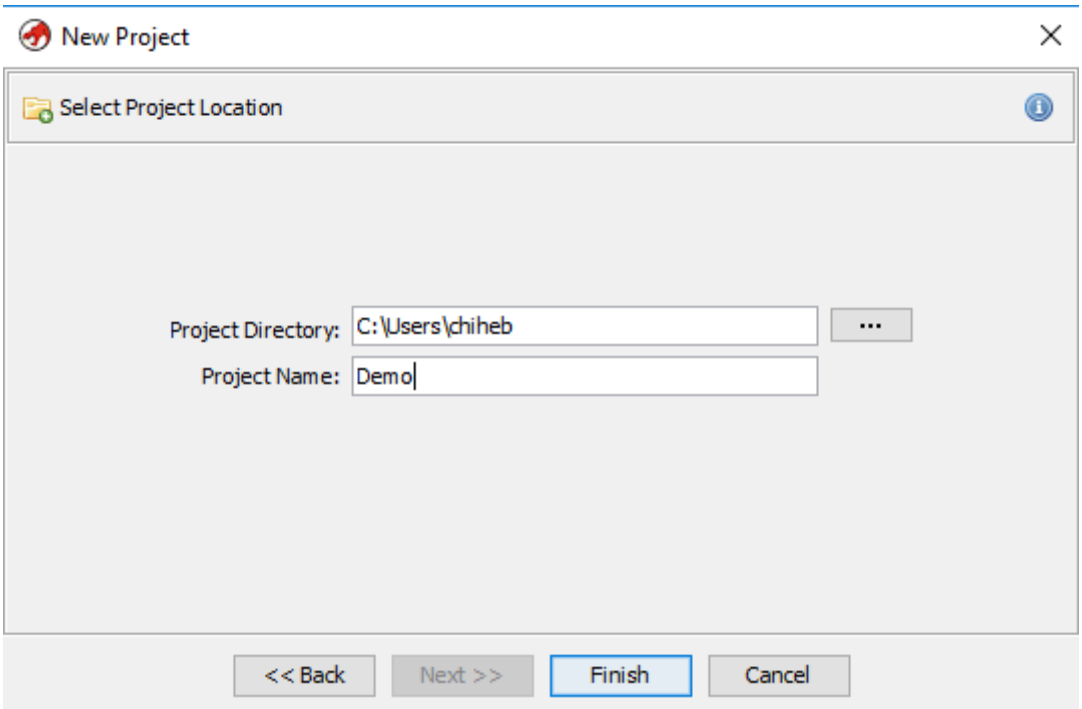
Open Ghidra



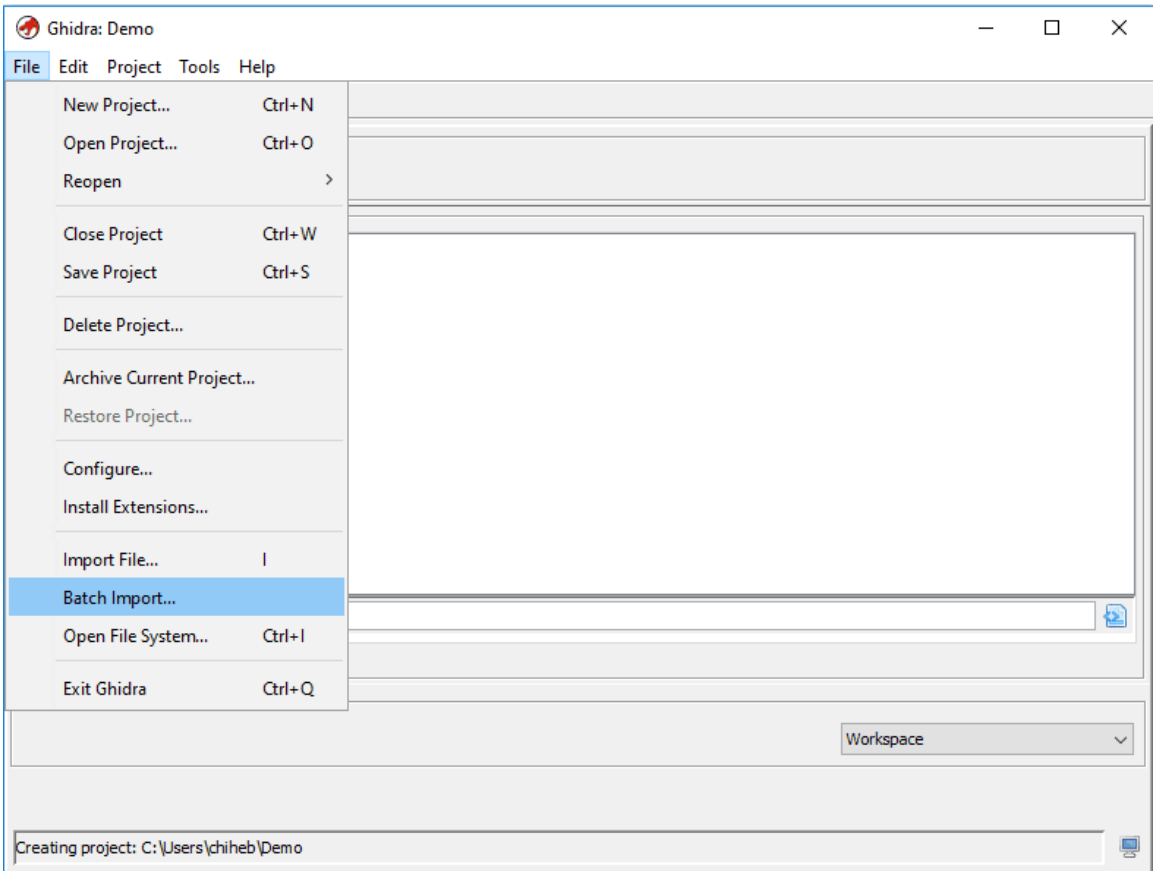
Start a new project:



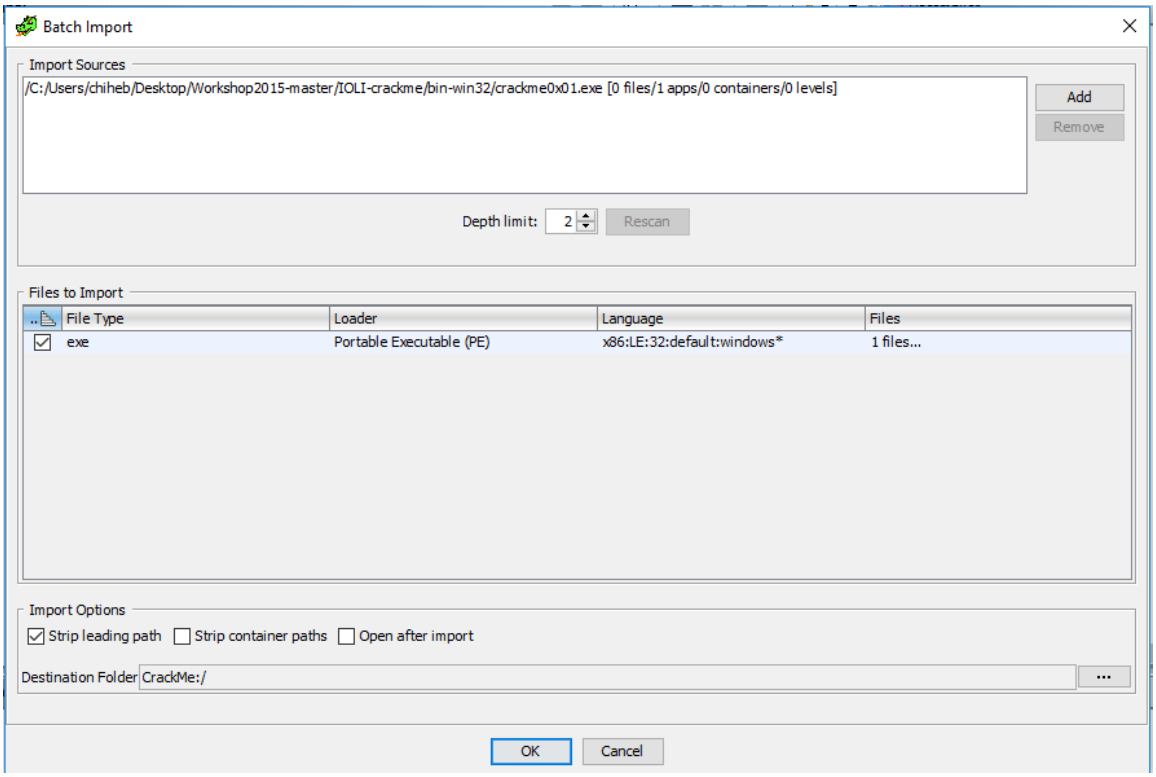
Name the project



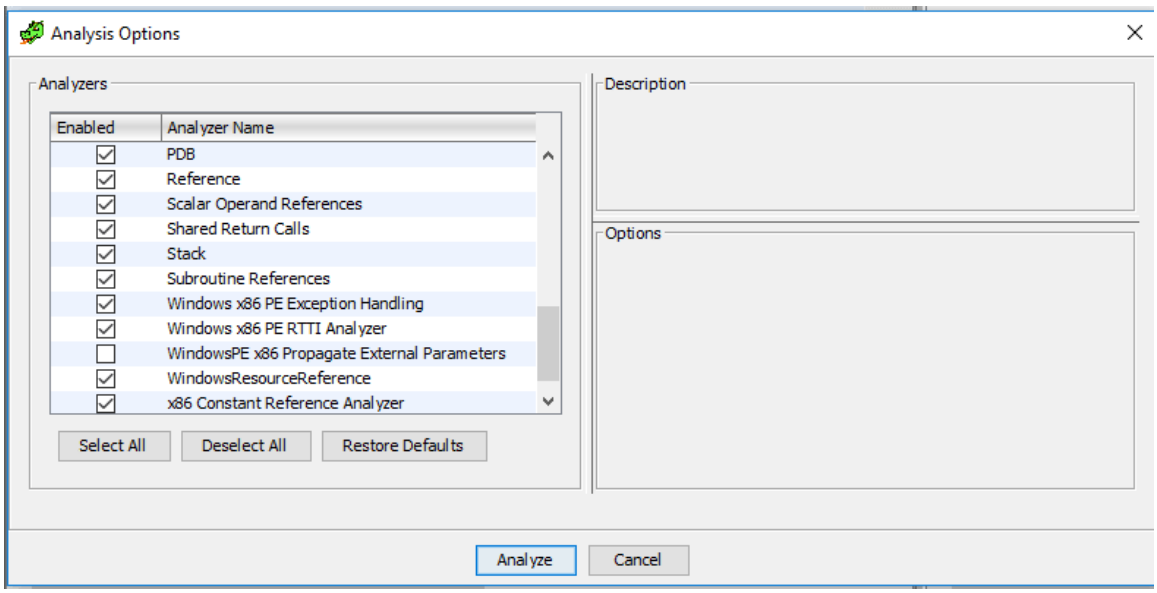
Import the binary with **Batch Import**



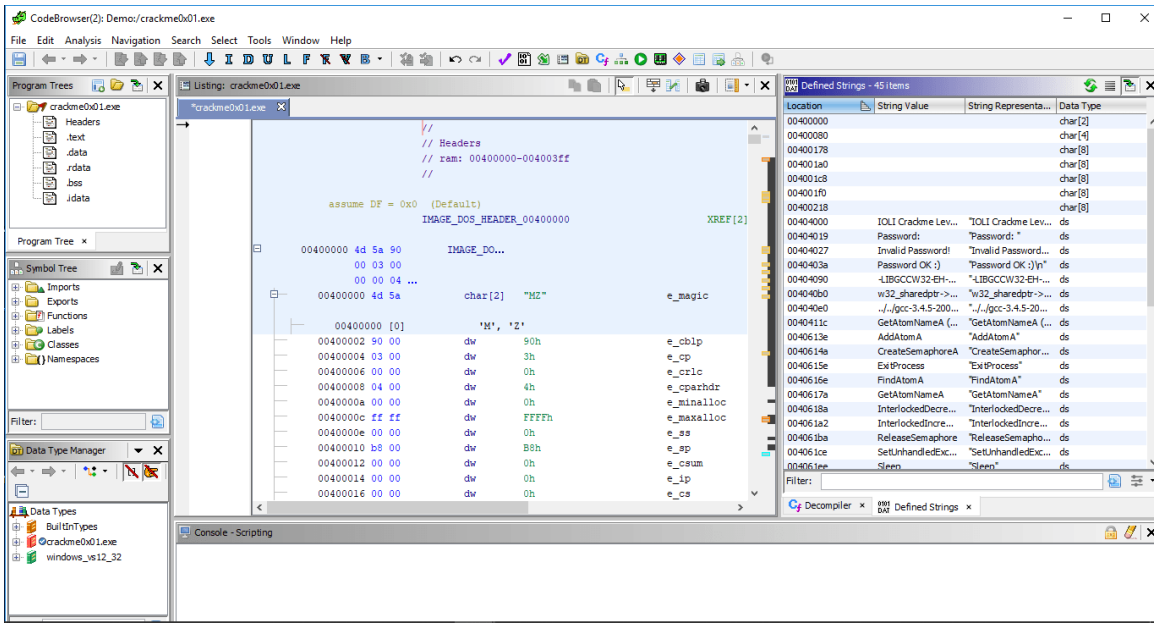
Open the binary



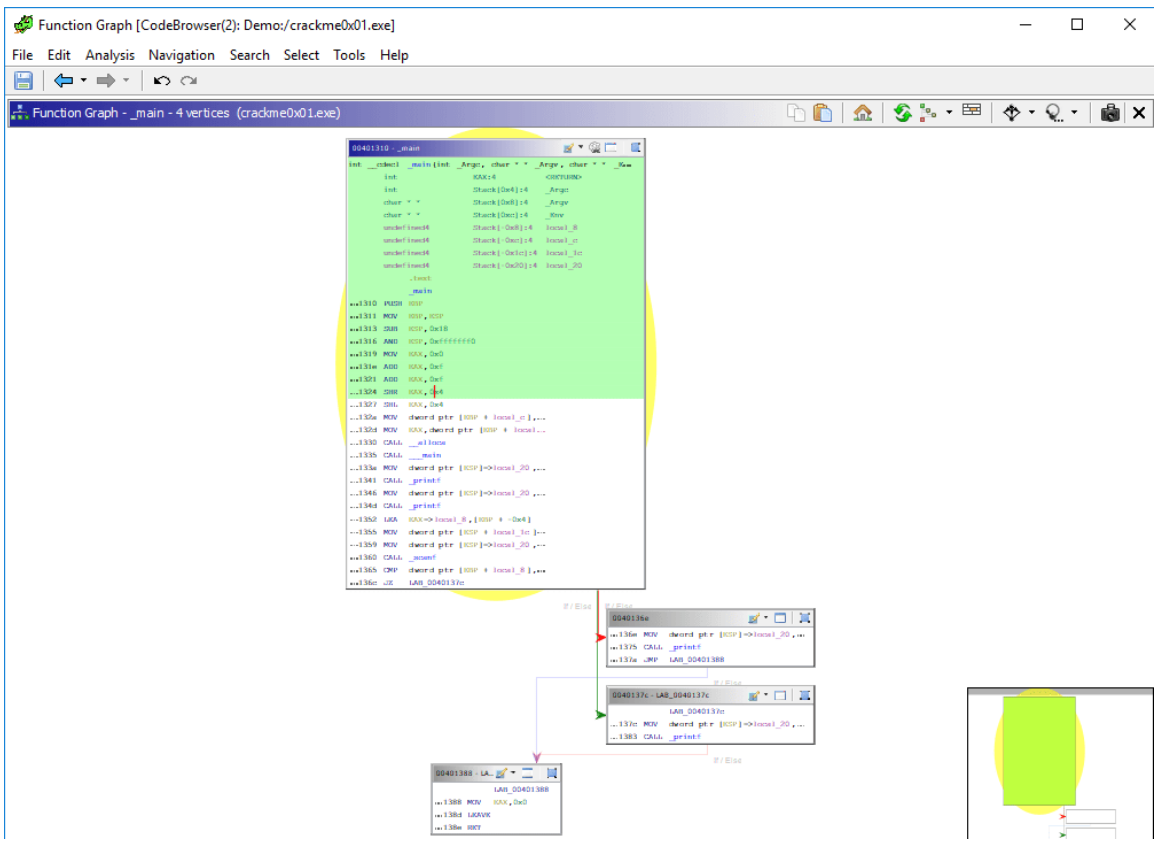
Select the required options and click "Analyze"



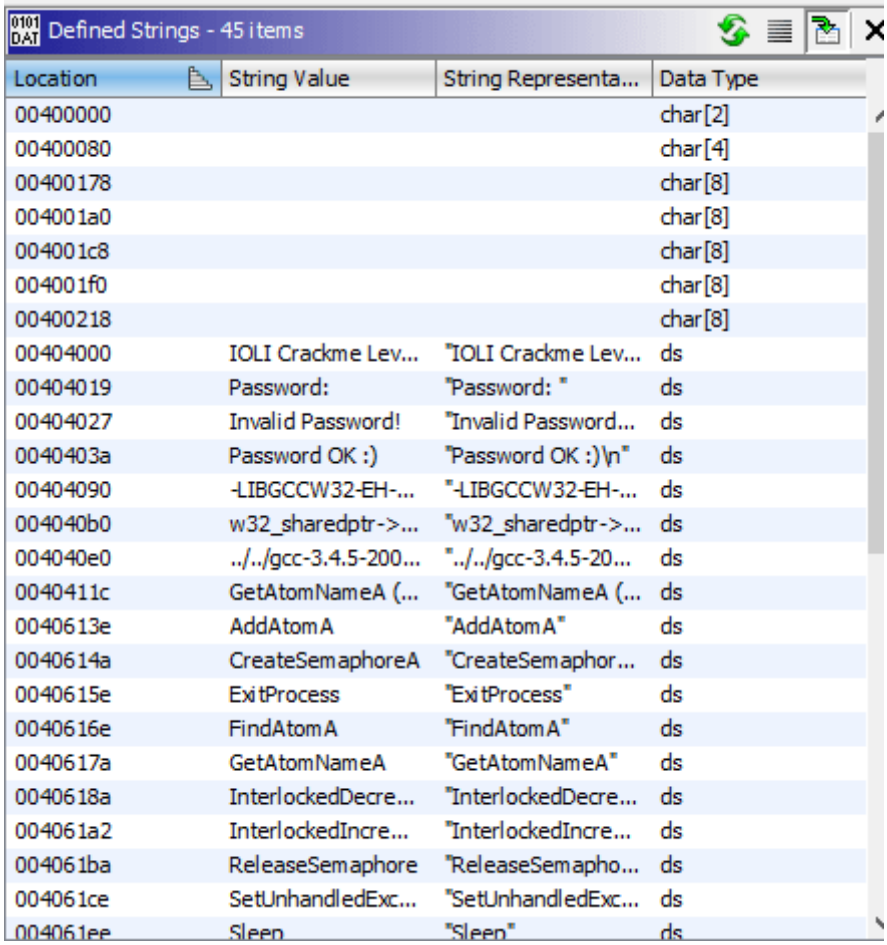
Voila! This is the main windows of Ghidra



You can also check the function graphs



To solve the challenge let's first start with extracting the binary strings



Defined Strings - 45 items

Location	String Value	String Representa...	Data Type
00400000			char[2]
00400080			char[4]
00400178			char[8]
004001a0			char[8]
004001c8			char[8]
004001f0			char[8]
00400218			char[8]
00404000	IOLI Crackme Lev...	"IOLI Crackme Lev...	ds
00404019	Password:	"Password: "	ds
00404027	Invalid Password!	"Invalid Password...	ds
0040403a	Password OK :)	"Password OK :)\n"	ds
00404090	-LIBGCCW32-EH-...	"-LIBGCCW32-EH-...	ds
004040b0	w32_sharedptr->...	"w32_sharedptr->...	ds
004040e0	../../gcc-3.4.5-200...	"../../gcc-3.4.5-20...	ds
0040411c	GetAtomNameA (...	"GetAtomNameA (...	ds
0040613e	AddAtomA	"AddAtomA"	ds
0040614a	CreateSemaphoreA	"CreateSemaphor...	ds
0040615e	ExitProcess	"ExitProcess"	ds
0040616e	FindAtomA	"FindAtomA"	ds
0040617a	GetAtomNameA	"GetAtomNameA"	ds
0040618a	InterlockedDecre...	"InterlockedDecre...	ds
004061a2	InterlockedIncre...	"InterlockedIncre...	ds
004061ba	ReleaseSemaphore	"ReleaseSemapho...	ds
004061ce	SetUnhandledExc...	"SetUnhandledExc...	ds
004061ee	Sleep	"Sleep"	ds

As you can notice we get all the strings of the file. One of them is "Password OK :)"

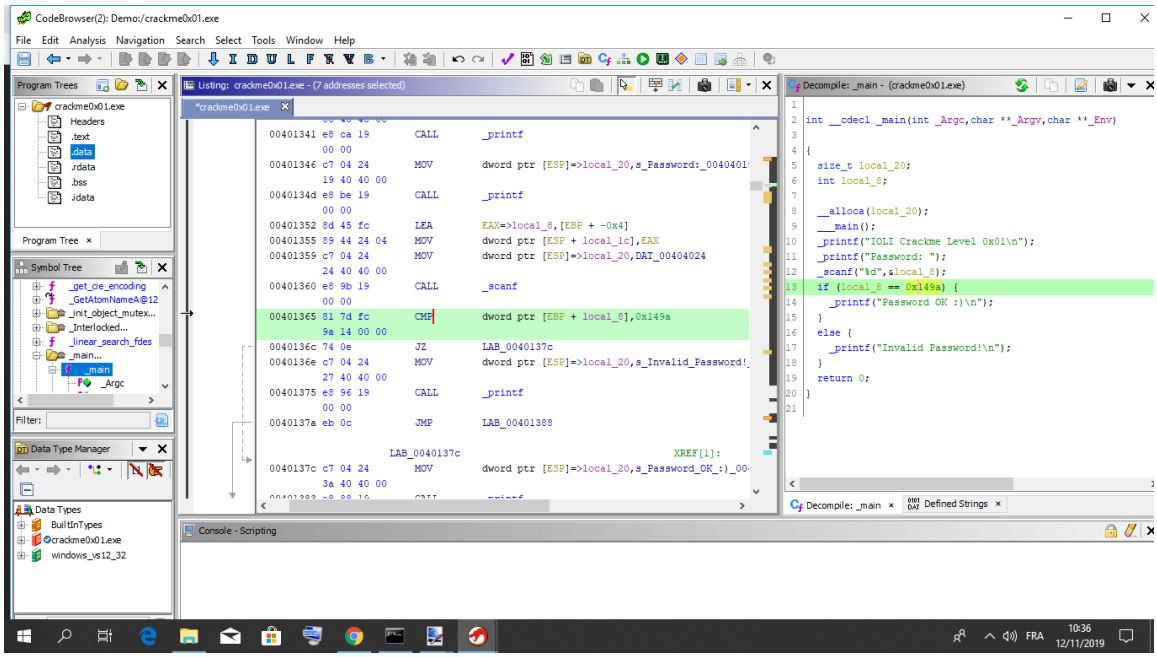
Ghidra is powerful. It gives you the ability to decompile the file. As you can see from the screenshot it is giving us a readable code.

If you check the code carefully you will notice this line of code

```
If (local_8 == 0x149a)

    _Printf ( “Password OK :) /n ”)
```

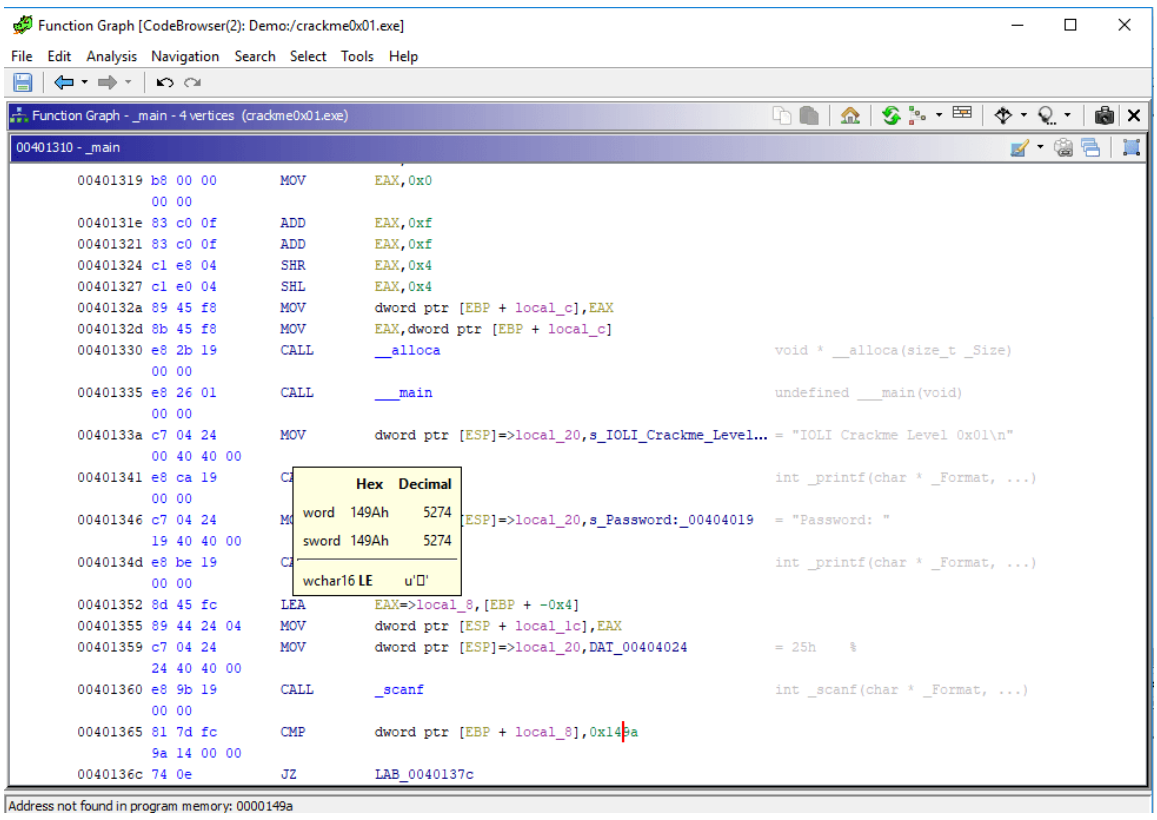
At the other side of the window you will see the CMP instruction. With a small Google search you will find that

"*CMP is generally used in conditional execution. This __ _* ***instruction _ basically subtracts one operand from the other for comparing whether the operands are equal or not. It does not disturb the destination or source operands. It is used along with the conditional jump _ instruction _ __*** *for decision making. "*



Then if our analysis is correct then the valid password will be a conversion of "0x149a"

To check its value double click on it and you will get this.



The decimal value is "5274". So let's try it:

Go back to your terminal and run the binary and this time type 5274:

```
C:\Users\chiheb\Desktop\Workshop2015-master\IOLI-crackme\bin-win32>crackme0x01.exe
IOLI Crackme Level 0x01
Password: 5274
Password OK :)
```

Congratulations, you solved your first **crackme** challenge.

This article will be updated with more interesting sections in the next few hours like Malware Analysis with Ghidra

Further resources

- https://ghidra-sre.org/CheatSheet.html

References

- https://www.tutorialspoint.com/assembly_programming/assembly_conditions.htm

Summary

This article was a good opportunity to learn the fundamentals of reverse engineering with an amazing tool called "Ghidra"

How to Perform Memory Analysis

How to Perform Memory Analysis

Source: malware-analysis-virtual-box-cyber-forensicator.jpg

Abstract

Malware threats are a very serious problem in information security nowadays. Dangerous hackers are inventing new techniques on a daily basis to bypass security layers and avoid detection. Thus it is time to figure out how to analyse memorydumps as.

But this time I want to take this opportunity to elaborate more Memory analysis because it is a required skill to every Forensics expert and malware analyst.

In this Article we are going to learn:

- **Dissecting Memory**
- **Memory Management**
- **Computer Forensic analysis steps**
- **Digital Evidence acquisition**
- **Memory Acquisition**
- **Memory Analysis**
- **Volatility Framework**
- **Memory Analysis Best Practices**

Memory Analysis

Malware analysis is the art of determining the functionality, origin and potential impact of a given malware sample, such as a virus, worm, trojan horse, rootkit, or backdoor. As a malware analyst, your main role is to collect all the information about the malicious software and have a good understanding of what happened to the infected machines. Like any process, to perform a malware analysis you typically need to follow a certain methodology and a number of steps.

Memory malware analysis is widely used for digital investigation and malware analysis. It refers to the act of analysing a dumped memory image from a targeted machine after executing the malware to obtain multiple numbers of artefacts including network information, running processes, API hooks, kernel loaded modules, Bash history, etc. ... This phase is very important because it is always a good idea to have a clearer understanding of the malware capabilities.

- Process list and the associated threads
- networking information and interfaces (TCP/UDP) • Kernel modules including the hidden modules
- Opened files in the kernel
- Bash and commands history
- System Calls • Kernel hooks

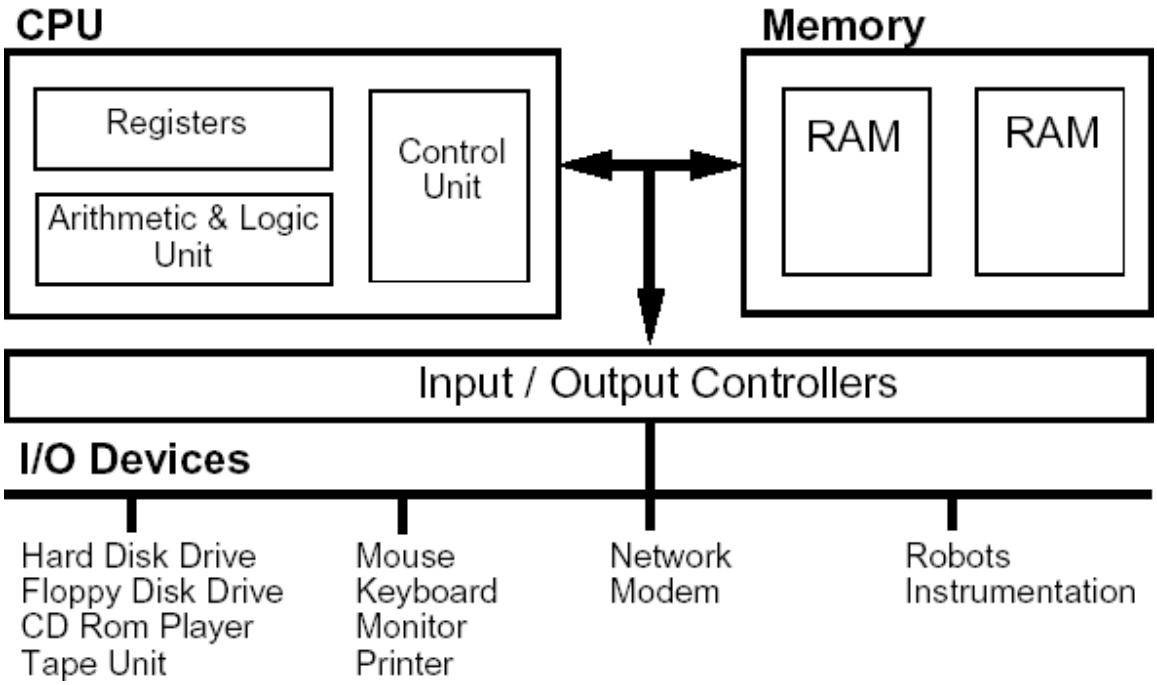
Dissecting Memory

If we are going to learn how to analyse memory dumps we need first to explore what memory is? and how it works.

Memory is a vital component in the computer architecture. Computers are composed by:

- **CPU**
- **Controllers**
- **Memory**

The full architecture is described in the following graph:



source overall.gif

In memory analysis, we are dealing with **RAM** s.

*A __**RAM (pronounced ramm) is an acronym for random access memory, a type of computer memory that can be accessed randomly; that is, any byte of memory can be accessed without touching the preceding bytes. RAM**__ is found in servers, PCs, tablets, smartphones and other devices_*, such as printers. __ **RAM is volatile** _

source: RAM061711.jpg

The memory is divided into 4,096-byte memory chunks named pages, to facilitate internal handling. The 12 least significant bits are the offset; the rest is the page number. On the recent x86 architecture, For example, the Linux kernel divides the virtual space, usually 4 GB into 3 GB dedicated to UserLand, and 1 GB for kernel land. This operation is named segmentation. The kernel uses a page table for the correspondence between physical and virtual addresses. To manage the different regions of memory, it uses a virtual memory area (VMA)

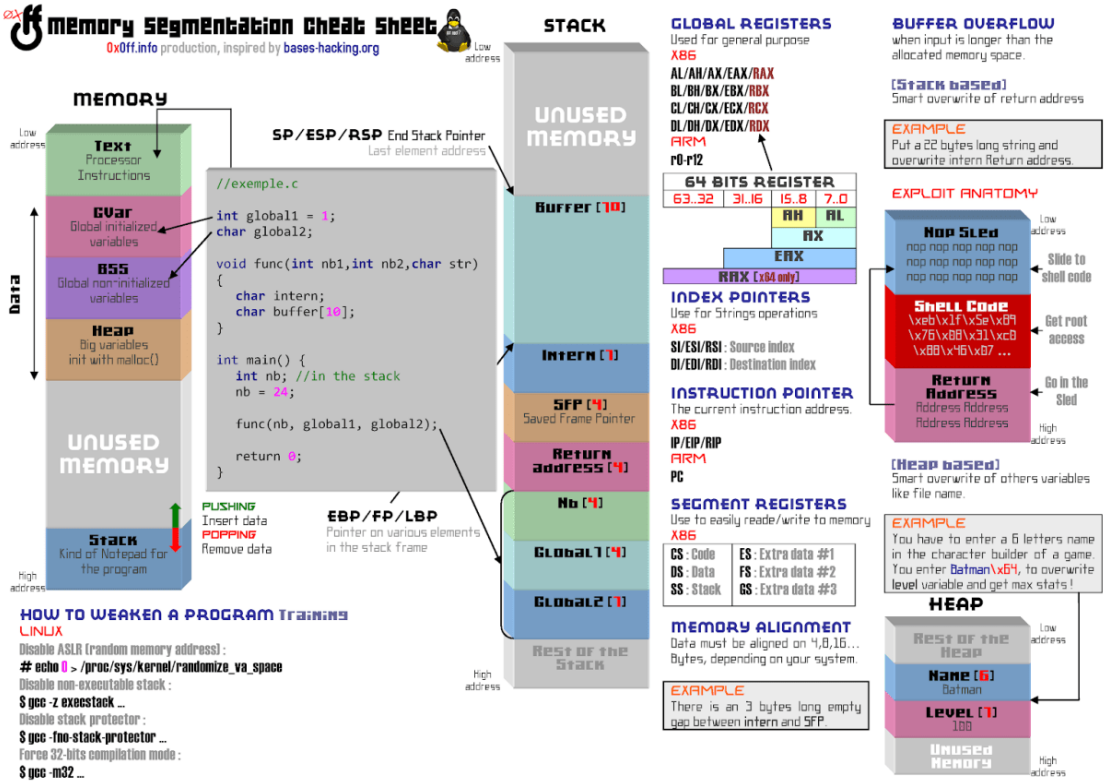
The stack is a special memory space. In programming, it is an abstract data type used to collect elements using two operations: push and pop. This section grows automatically, but

when it becomes closer to another memory section, it will cause a problem and a confusion to the system. That is why attackers are using this technique to confuse the system with other memory areas.

The heap is used for dynamic memory allocation. It resides in the RAM like the stack, but it is slower. The kernel heap is using the following three types of allocators:

- **SLAB:** This is a cache-friendly allocator.
- **A simple list of blocks (SLOB)**: This is an allocator used in small systems. It uses a first-fit algorithm.
- **SLUB** : It is the default Linux allocator.

You can explore the detailed sections of memory check this great cheat sheet:



Better resolution here Memory Segmentation sheet

Memory Management

Memory management is an important capability of every operating system. It is also integrated into Linux kernel. Linux manages memory in a virtual way. In other words, there is no correspondence between the physical memory addresses, and the addresses used and seen by the program. This technique gives the users and developers flexibility. Linux is dealing with the following five types of addresses:

1. User virtual addresses

2. Physical addresses
3. Bus addresses
4. Kernel logical addresses
5. Kernel virtual addresses

Computer Forensic analysis steps

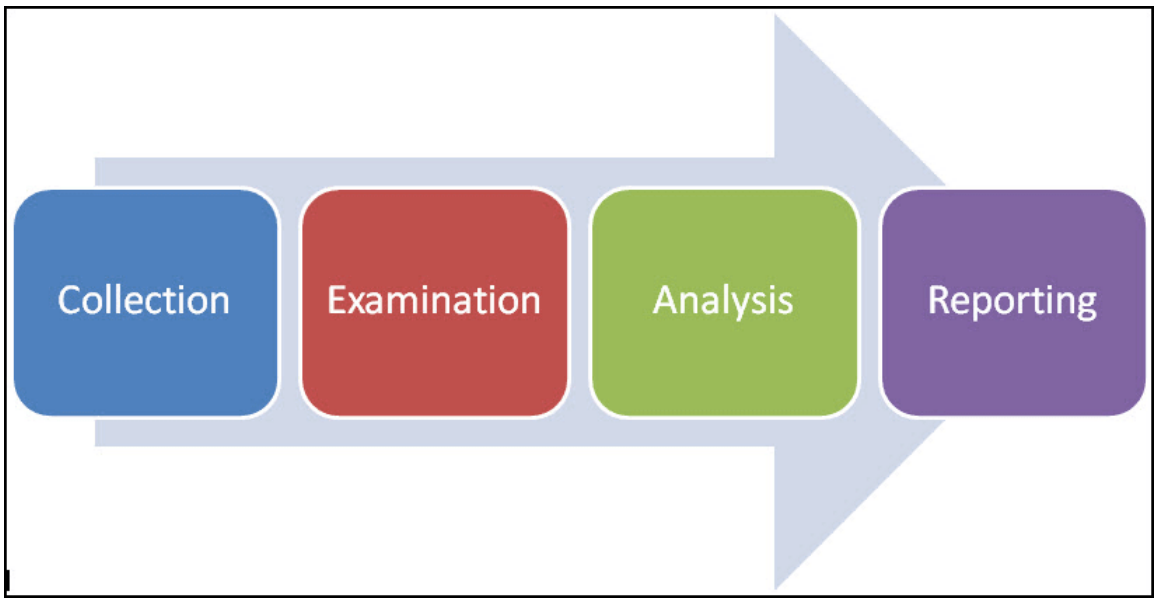
NIST is describing Forensics as the following:

*The most common goal of performing forensics is to gain a better understanding of an event of interest by finding and analyzing the facts related to that event... Forensics may be needed in many different situations, such as evidence collection for legal_* proceedings and internal disciplinary actions, and handling of malware incidents and unusual operational problems. _

Like any methodological operation, Computer forensic analysis goes through well-defined steps: Collection; Examination, Analysis and reporting. let's explore these steps one by one:

1. **Collection:** identifying data sources and verify the integrity of it
2. **Examination:** assessing and extracting the relevant pieces of information from the collected data
3. **Analysis**
4. **Reporting**

The steps are based on the **NIST Guide to Integrating Forensic Techniques into Incident Response**. I highly recommend exploring the Process in details (Performing the Forensic Process)



source: nist+process.jpg

Digital Evidence acquisition

Digital evidence needs to be treated carefully because we are going to analyse them. Also, we need to use them later within the legal process. Eliézer Pereira prioritized them in his Article RAM Memory Forensic Analysis as the following from the most volatile to the least volatile:

- Caches
- Routing tables, process tables, memory
- Temporary system files
- Hard drive
- Remote logs, monitoring data
- Physical network configuration, network topology
- Media files (CDs, DVDs)

Memory Acquisition

The first step of memory analysis is memory acquisition by dumping the memory of a machine using a number of utilities. One of these tools is fmem, which is a kernel module to create a new device called /dev/fmem to allow direct access to the whole memory. After downloading it from their official repository and compiling it you can acquire the machine memory using this command:

```
# dd if=/dev/fmem of=... bs=1MB count=...
```

Another tool is The Linux Memory Extractor. LIME is a Loadable Kernel Module (LKM) to allow volatile memory acquisition from Linux and Linux- based devices, such as Android.

These are some free Memory Acquisition tools:

- WindowsSCOPE
- https://belkasoft.com/ram-capture
- **winen**
- **Mdd (Memory DD)** (is no longer under active development.)
- HBGary

A full list of useful tools can be found here: **Tools: Memory Imaging (https://www.forensicswiki.org/wiki/Tools:Memory_Imaging)**

After having a memory dump, it is time to analyze the memory image.

Memory Analysis with Volatility Framework

To analyse memory You can simply use volatility framework, which is an open source memory forensics tool written in Python. It is available under GPL. Volatility comes with various plugins and a number of profiles to ease obtaining basic forensic information about memory image files. To download it you can visit this website: The Volatility Foundation - Open Source Memory Forensics or GitHub - volatilityfoundation/volatility

```
BEATRIX volatility # python vol.py sockets -f /home/michael/stuxnet.vmem
Volatility Foundation Volatility Framework 2.3.1
Offset(V)       PID   Port  Proto Protocol        Address         Create Time
---------- -------- ------ ------ -------------- --------------- -----------
0x81dc2008      680    500     17 UDP            0.0.0.0         2010-10-29 17:09:05 UTC+0000
0x82061c08        4    445      6 TCP            0.0.0.0         2010-10-29 17:08:53 UTC+0000
0x82294aa8      940    135      6 TCP            0.0.0.0         2010-10-29 17:08:55 UTC+0000
0x821a5008      188   1025      6 TCP            127.0.0.1       2010-10-29 17:09:09 UTC+0000
0x81cb3d70     1080   1141     17 UDP            0.0.0.0         2010-10-31 16:36:16 UTC+0000
0x81da4d18      680      0    255 Reserved       0.0.0.0         2010-10-29 17:09:05 UTC+0000
0x81fdbe98     1032    123     17 UDP            127.0.0.1       2011-06-03 04:25:47 UTC+0000
0x81c79778     1080   1142     17 UDP            0.0.0.0         2010-10-31 16:36:16 UTC+0000
0x81c20898     1200   1900     17 UDP            127.0.0.1       2011-06-03 04:25:47 UTC+0000
0x82060008      680   4500     17 UDP            0.0.0.0         2010-10-29 17:09:05 UTC+0000
0x81cb9e98     1580   5152      6 TCP            127.0.0.1       2010-10-29 17:09:05 UTC+0000
0x81da54b0        4    445     17 UDP            0.0.0.0         2010-10-29 17:08:53 UTC+0000
```

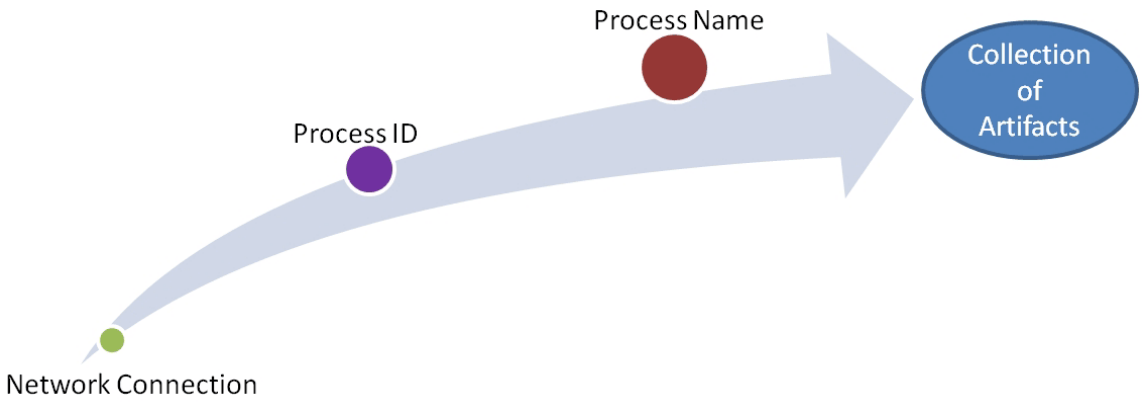
source: volatility-sockets.gif

To identify malicious network activities many experts recommend following these steps. First, you can identify Process IDs of network connections.

```
root@siftworkstation:/home/sansforensics/Desktop/shared# vol.py --profile=Win2003SP1x86 pslist -f CMIT_460_Lab_3-4.vmem
Volatility Foundation Volatility Framework 2.4
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ----------
0x8659f2a8 System                    4      0     65      565 ------      0
0x8605a838 smss.exe                428      4      3       18 ------      0 2009-10-30 17:15:29 UTC+0000
0x864ada30 csrss.exe               692    428     12      465      0      0 2009-10-30 17:15:43 UTC+0000
0x8616eb18 winlogon.exe            820    428     18      511      0      0 2009-10-30 17:15:49 UTC+0000
0x85ff19a0 services.exe           1016    820     16      301      0      0 2009-10-30 17:15:59 UTC+0000
0x863096d0 lsass.exe              1060    820     27      444      0      0 2009-10-30 17:16:01 UTC+0000
0x85fec940 svchost.exe            1432   1016      6       80      0      0 2009-10-30 17:16:14 UTC+0000
0x86301cd8 svchost.exe            1664   1016     11      232      0      0 2009-10-30 17:16:22 UTC+0000
0x863b6538 svchost.exe            1748   1016      9      137      0      0 2009-10-30 17:16:23 UTC+0000
0x8615cd88 svchost.exe            1824   1016     13      158      0      0 2009-10-30 17:16:24 UTC+0000
0x86122ae8 svchost.exe            1848   1016     43      900      0      0 2009-10-30 17:16:25 UTC+0000
0x8648d538 explorer.exe            880    856     10      329      0      0 2009-10-30 17:16:55 UTC+0000
0x86113d88 sqlmangr.exe           1512    880      2       73      0      0 2009-10-30 17:17:13 UTC+0000
0x860f52b8 spoolsv.exe             472   1016     14      134      0      0 2009-10-30 17:17:35 UTC+0000
0x8648cc40 msdtc.exe               508   1016     21      158      0      0 2009-10-30 17:17:35 UTC+0000
0x8616a6b8 svchost.exe             744   1016      2       56      0      0 2009-10-30 17:17:40 UTC+0000
0x8612f020 inetinfo.exe            816   1016      9      183      0      0 2009-10-30 17:17:41 UTC+0000
0x8610e020 sqlservr.exe            876   1016     22      251      0      0 2009-10-30 17:17:41 UTC+0000
0x85d3ed88 svchost.exe            1240   1016      2       40      0      0 2009-10-30 17:17:47 UTC+0000
0x86022d88 surveyor.exe           1284   1016      7      109      0      0 2009-10-30 17:17:47 UTC+0000
0x86017898 svchost.exe            1812   1016     16      153      0      0 2009-10-30 17:17:50 UTC+0000
0x85d248a0 sqlagent.EXE            336   1016      8      135      0      0 2009-10-30 17:17:55 UTC+0000
0x85d2f020 svchost.exe             916   1016     15      133      0      0 2009-10-30 17:18:01 UTC+0000
0x85cfa020 wmiprvse.exe           2428   1432      4      168      0      0 2009-10-30 17:18:22 UTC+0000
0x85cacd88 cmd.exe                4024   1848      1       32      0      0 2009-10-30 17:51:00 UTC+0000
0x85d377a8 tango.exe              3632   4024      1      139      0      0 2009-10-30 17:55:23 UTC+0000
0x85cab510 svchost.exe            2692   3632      1      128      0      0 2009-10-30 17:59:11 UTC+0000
0x85cd7d88 w3wp.exe               3796   1812     13      121      0      0 2009-10-30 18:14:22 UTC+0000
0x85cc6910 wmiprvse.exe           1244   1432      6      121      0      0 2009-10-30 18:17:15 UTC+0000
0x85cc9b18 iexplore.exe           3280   1220     19       95      0      0 2009-10-30 18:17:15 UTC+0000
root@siftworkstation:/home/sansforensics/Desktop/shared#
```

source: pslist.png

Later you need to map that IDs to Process Names and later terminate every step and process by collecting the artefacts by taking notes, screenshots and of course time-stamps.



source: Fig2lg061711.jpg

Note: this section is not completed yet. The processes will be described in a detailed way. Stay tuned.

Peerlyst Articles about Memory Analysis you need to explore

- Useful PhD thesis: Advances in Modern Malware and Memory Analysis - contains 4 new proposals
- Some useful forensics tools for your forensics investigation
- How to build a Linux Automated Malware Analysis Lab
- LiME: Loadable Kernel Module Overview
- Malware analysis Frameworks
- Memory Forensics : Tracking Process Injection

Summary

In this article, we explored how to perform Malware memory analysis.

Post Updates

Checked the availability of tools (Thanks to **Ken Pryor**)

References

1. https://technical.nttsecurity.com/post/102egyy/hunting-malware-with-memory-analysis
2. Advanced Infrastructure Penetration Testing Chiheb Chebbi
3. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-86.pdf
4. https://www.cybrary.it/0p3n/ram-memory-forensic-analysis/

5. https://resources.infosecinstitute.com/memory-forensics/#gref
6. https://technical.nttsecurity.com/post/102egyy/hunting-malware-with-memory-analysis
7. What is RAM - Random Access Memory? Webopedia Definition

Red Teaming Attack Simulation with "Atomic Red Team"

Modern organizations face cyber threats on a daily basis. Black hat hackers do not show any indication that they are going to stop. New hacking techniques appear regularly. According to multiple information security reports, the number of APT attacks is increasing in a notable way, targeting national defenses, manufacturing, and the financial industry. Thus, classic protection techniques are, in many cases, useless. Deploying suitable platforms and solutions can help organizations and companies defend against cyber attacks, especially APTs. Some of these platforms are attack simulation tools. In this article we are going to learn how to deploy a red teaming simulation platform called **Atomic Red Team**

But first what is Red teaming?

Techtarget defines red teaming as follows:

> “Red teaming is the practice of rigorously challenging plans, policies, systems and assumptions by adopting an adversarial approach. A red team may be a contracted external party or an internal group that uses strategies to encourage an outsider perspective.”

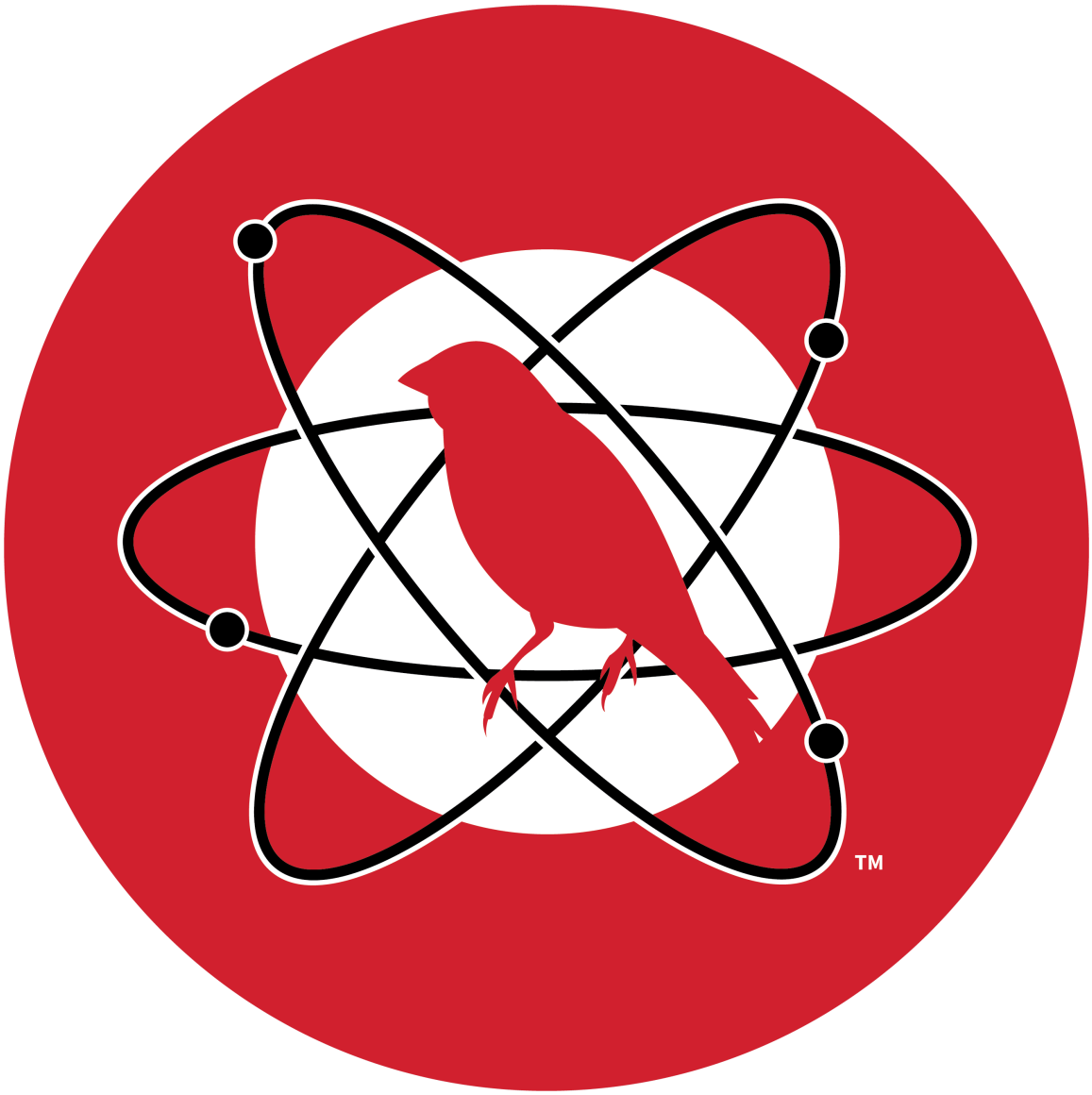
Red Teamers usually perform the following steps:

- Recon
- Initial compromise
- Establish persistence
- Escalate privileges
- Internal Recon
- Lateral movement
- Data analysis
- Exfiltrate and complete mission

Image source

Atomic Red Team

According to its official Github repository



> *Atomic Red Team allows every security team to test their controls by executing simple "atomic tests" that exercise the same techniques used by adversaries (all mapped to Mitre's ATT&CK). Atomic Red Team is a library of simple tests that every security team can execute to test their controls. Tests are focused, have few dependencies, and are defined in a structured format that can be used by automation frameworks.*

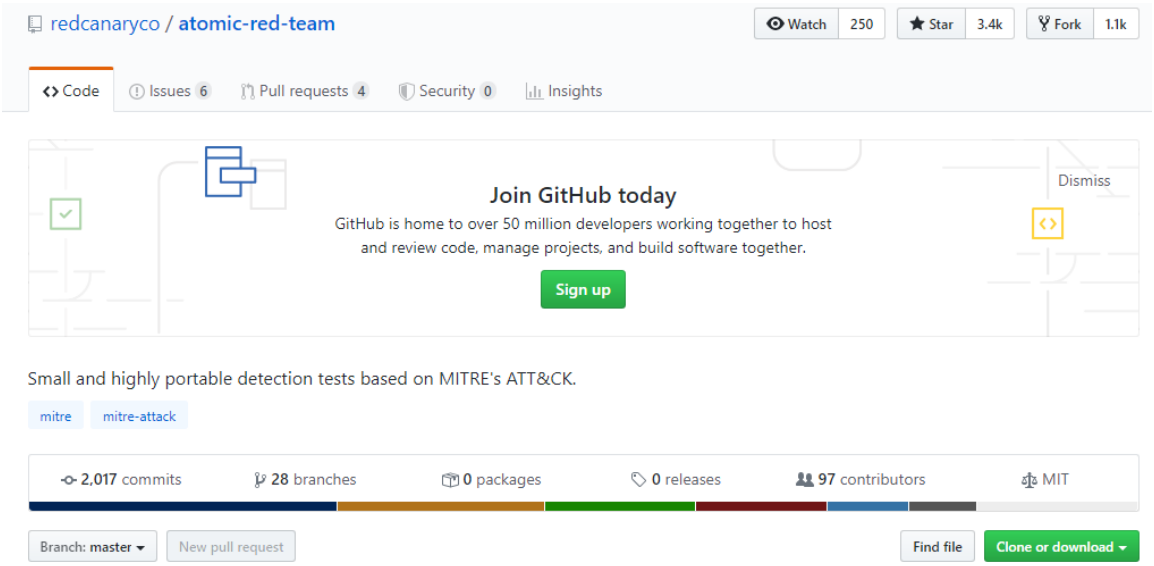
MITRE ATT&CK is a framework developed by the Mitre Corporation. The comprehensive document classifies adversary attacks, in other words, their techniques and tactics after observing millions of real-world attacks against many different organizations. This is why ATT&CK refers to "Adversarial Tactics, Techniques & Common Knowledge". A tactic is the highest level of attack behaviour. Techniques are used to execute an attack successfully

MITRE framework present the tactics as the following:

1. **Initial Access**
2. **Execution**
3. **Persistence**
4. **Privilege Escalation**
5. **Defense Evasion**
6. **Credential Access**
7. **Discovery**
8. **Lateral Movement**
9. **Collection**
10. **Exfiltration**
11. **Command and Control**

Let's explore how to install and use Atomic Red Team:

First you need to download the project from here: https://github.com/redcanaryco/atomic-red-team



Disable Windows defender

Extract the zip file:

```
azureuser@OSQuery:~$ wget https://github.com/kolide/fleet/releases/latest/download/fleet.zip
--2020-06-10 07:27:36--  https://github.com/kolide/fleet/releases/latest/download/fleet.zip
Resolving github.com (github.com)... 140.82.118.3
Connecting to github.com (github.com)|140.82.118.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github.com/kolide/fleet/releases/download/2.6.0/fleet.zip [following]
--2020-06-10 07:27:36--  https://github.com/kolide/fleet/releases/download/2.6.0/fleet.zip
Reusing existing connection to github.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/64099814/54aa1a00-
6dba-11ea-9324-6a129cdd3148?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CS
VEH53A%2F20200610%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200610T072737Z&X-Amz-Expires=
300&X-Amz-Signature=1750ac9a1ff866c2cc30c51c5d08e50e6a51169d6e61acfcea1a514b61427e0f&X-Amz-S
ignedHeaders=host&actor_id=0&repo_id=64099814&response-content-disposition=attachment%3B%20f
ilename%3Dfleet.zip&response-content-type=application%2Foctet-stream [following]
--2020-06-10 07:27:37--  https://github-production-release-asset-2e65be.s3.amazonaws.com/640
99814/54aa1a00-6dba-11ea-9324-6a129cdd3148?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential
=AKIAIWNJYAX4CSVEH53A%2F20200610%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200610T072737Z
&X-Amz-Expires=300&X-Amz-Signature=1750ac9a1ff866c2cc30c51c5d08e50e6a51169d6e61acfcea1a514b6
1427e0f&X-Amz-SignedHeaders=host&actor_id=0&repo_id=64099814&response-content-disposition=at
tachment%3B%20filename%3Dfleet.zip&response-content-type=application%2Foctet-stream
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release
-asset-2e65be.s3.amazonaws.com)... 52.216.230.139
Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-production-rel
```

The techniques can be found in the "atomics" folder:

Name	Date modified	Type	Size
.circleci	5/26/2020 11:44 AM	File folder	
.github	5/26/2020 11:44 AM	File folder	
ARTifacts	5/26/2020 11:44 AM	File folder	
atomic_red_team	5/26/2020 11:44 AM	File folder	
atomics	5/26/2020 11:44 AM	File folder	
bin	5/26/2020 11:44 AM	File folder	
docs	5/26/2020 11:44 AM	File folder	
execution-frameworks	5/26/2020 11:44 AM	File folder	
.gitignore	5/26/2020 11:44 AM	GITIGNORE File	1 KB
atomic-red-team.gemspec	5/26/2020 11:44 AM	GEMSPEC File	1 KB
CODE_OF_CONDUCT.md	5/26/2020 11:44 AM	MD File	4 KB
Gemfile	5/26/2020 11:44 AM	File	1 KB
Gemfile.lock	5/26/2020 11:44 AM	LOCK File	7 KB
LICENSE.txt	5/26/2020 11:44 AM	Text Document	2 KB
README.md	5/26/2020 11:44 AM	MD File	4 KB

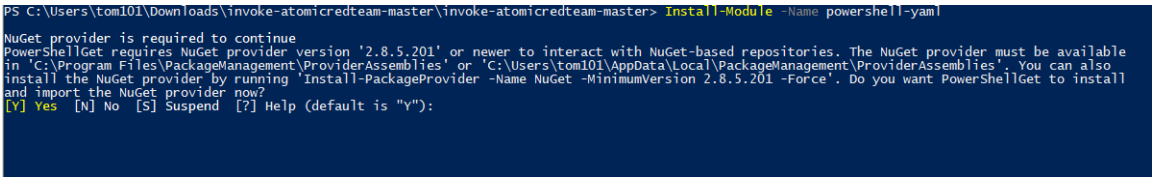
Now Open powershell and type:

```
powershell -ExecutionPolicy bypass
```

```
PS C:\Users\tom101\Downloads\invoke-atomicredteam-master\invoke-atomicredteam-master> ls


    Directory: C:\Users\tom101\Downloads\invoke-atomicredteam-master\invoke-atomicredteam-master


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         5/15/2020  11:16 AM                Private
d-----         5/15/2020  11:16 AM                Public
------         5/15/2020  11:16 AM           3222 CODE_OF_CONDUCT.md
------         5/15/2020  11:16 AM           4752 install-atomicredteam.ps1
------         5/15/2020  11:16 AM           3544 install-atomicsfolder.ps1
------         5/15/2020  11:16 AM           4928 Invoke-AtomicRedTeam.psd1
------         5/15/2020  11:16 AM            672 Invoke-AtomicRedTeam.psm1
------         5/15/2020  11:16 AM           1078 LICENSE.txt
------         5/15/2020  11:16 AM           1234 README.md


PS C:\Users\tom101\Downloads\invoke-atomicredteam-master\invoke-atomicredteam-master> powershell -ExecutionPolicy bypass
```

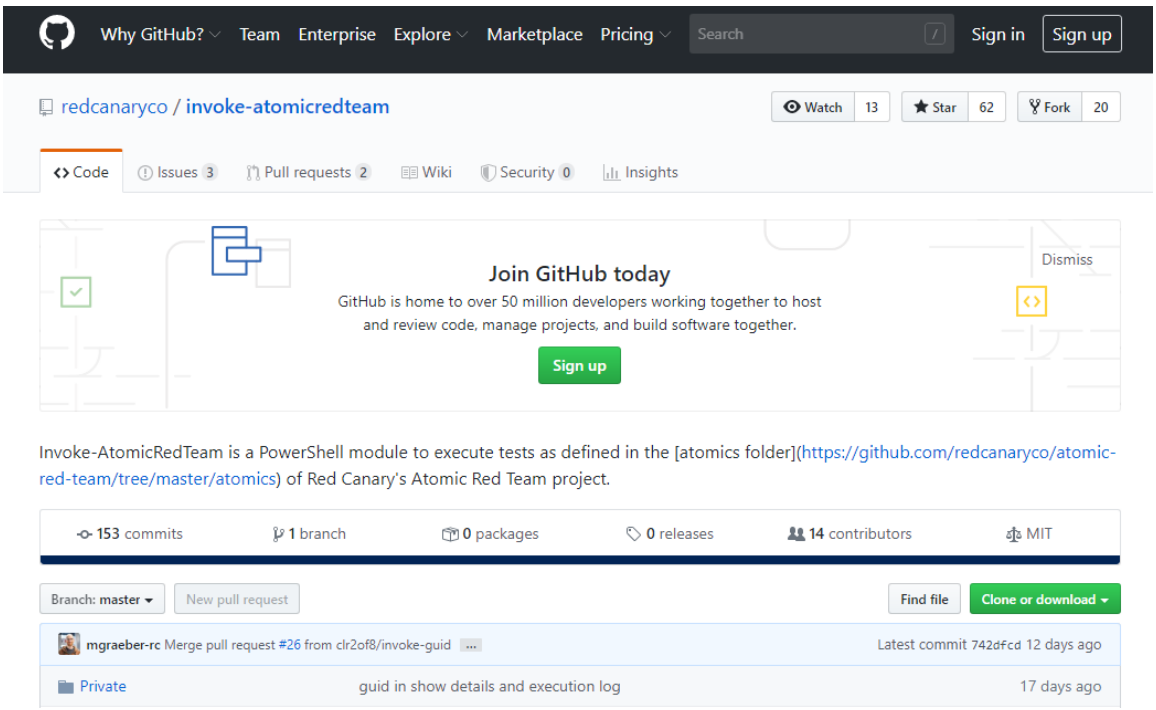
Install a required module:

```
Install-Module -Name powershell-yaml
```



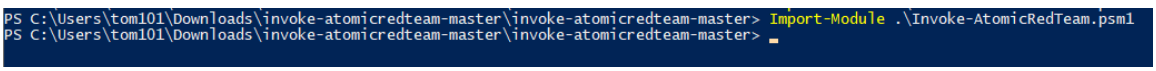
Now go and download **Invoke-atomicreadteam** from: https://github.com/redcanaryco/invoke-atomicredteam

*Invoke-AtomicRedTeam is a PowerShell module to execute __tests__ as defined in the atomics folder of Red Canary's Atomic Red __Team__ project. The "atomics folder" contains a folder for each __Technique__ defined by the MITRE ATT&CK™ Framework. Inside of each of these "T#" folders you'll find a __**yaml file that defines the attack __procedures** for each atomic test as well as an easier to read markdown (__**md) __version** of the same data.*



Enter the project folder and then type:

```
Import-Module ./Invoke-AtomicRedTeam.psm1
```

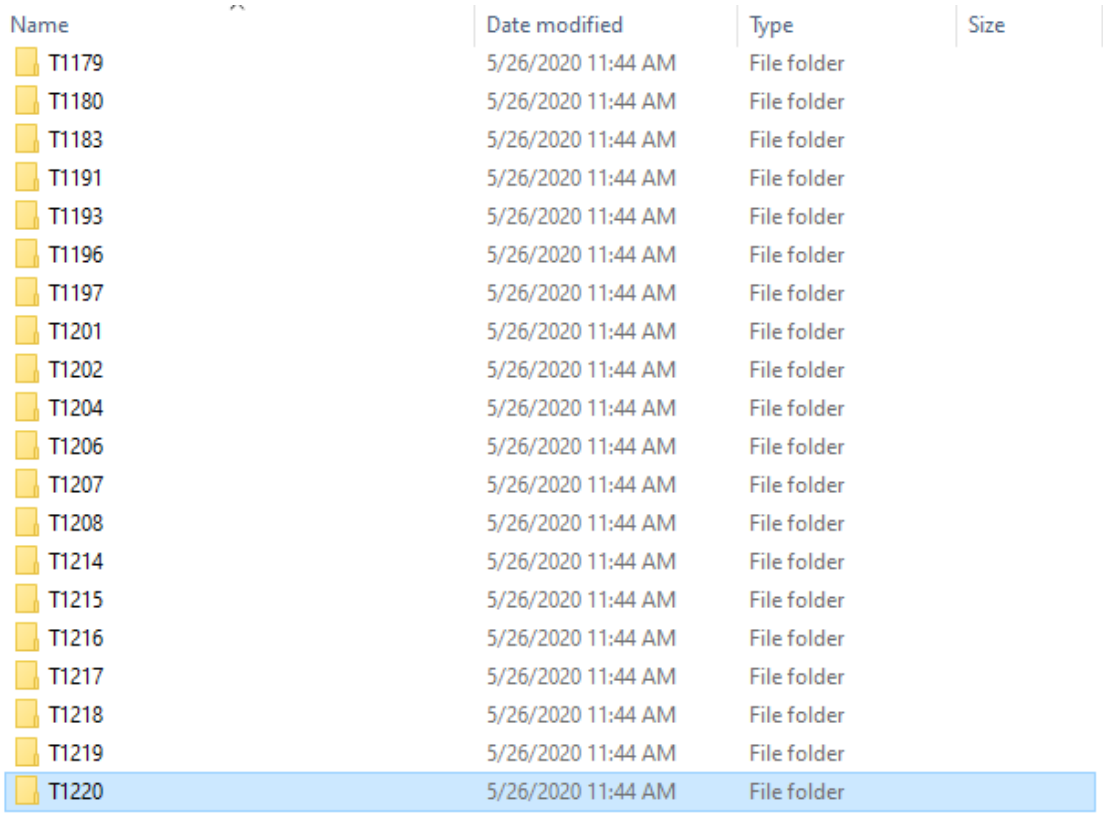


Now you can run any test you want by simply run the following commands:

```
$TXXXX = Get-AtomicTechnique -Path \path\to\atomics\TXXXX\TXXXX.yaml
```

```
Invoke-AtomicTest $TXXXX
```

The techniques can be found in the first downloaded project



Name	Date modified	Type	Size
T1179	5/26/2020 11:44 AM	File folder	
T1180	5/26/2020 11:44 AM	File folder	
T1183	5/26/2020 11:44 AM	File folder	
T1191	5/26/2020 11:44 AM	File folder	
T1193	5/26/2020 11:44 AM	File folder	
T1196	5/26/2020 11:44 AM	File folder	
T1197	5/26/2020 11:44 AM	File folder	
T1201	5/26/2020 11:44 AM	File folder	
T1202	5/26/2020 11:44 AM	File folder	
T1204	5/26/2020 11:44 AM	File folder	
T1206	5/26/2020 11:44 AM	File folder	
T1207	5/26/2020 11:44 AM	File folder	
T1208	5/26/2020 11:44 AM	File folder	
T1214	5/26/2020 11:44 AM	File folder	
T1215	5/26/2020 11:44 AM	File folder	
T1216	5/26/2020 11:44 AM	File folder	
T1217	5/26/2020 11:44 AM	File folder	
T1218	5/26/2020 11:44 AM	File folder	
T1219	5/26/2020 11:44 AM	File folder	
T1220	5/26/2020 11:44 AM	File folder	

References:

- https://bestestredteam.com/2019/07/30/atomic-red-team/
- https://bleepsec.com/2018/11/26/using-attack-atomic-red-team-part1.html

How to build a Machine Learning Intrusion Detection system

Introduction

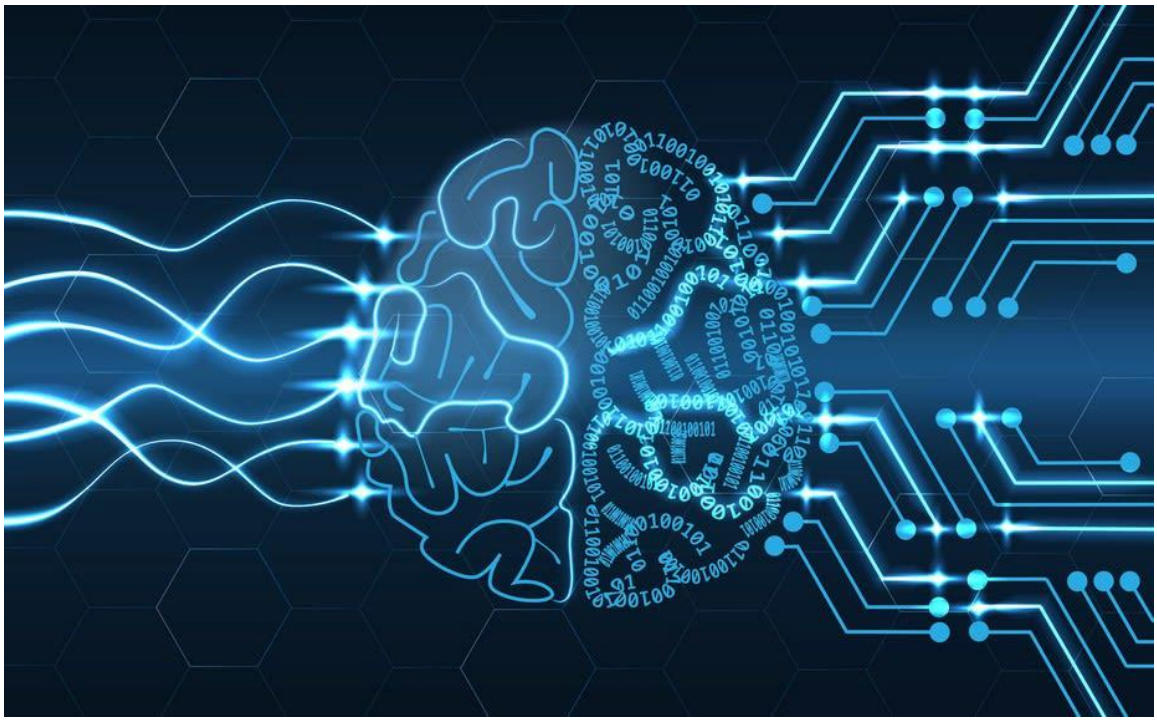
Machine learning techniques are changing our view of the world and they are impacting all aspects of our daily life. Thus machine learning is playing a huge role in information security. In this module you will not only explore the fundamentals behind machine learning techniques but you will dive into a hands-on experience to learn how to build real world Intrusion detection systems from scratch using cutting edge techniques, programming libraries and publicly available datasets.

This module will cover:

- Machine learning models
- The required steps to build a Machine learning project
- How to evaluate a machine learning Model
- Most useful Data Science and Machine learning libraries
- Artificial Neural Networks and Deep Learning
- Next Generation Intrusion detection systems using Machine learning Techniques.

Artificial intelligence

Artificial intelligence is the art of making computer programs to behave like a human and by behave i mean perceiving, learning, understanding and knowing. AI is involving many areas such as computer science, neuroscience, psychology and so on.



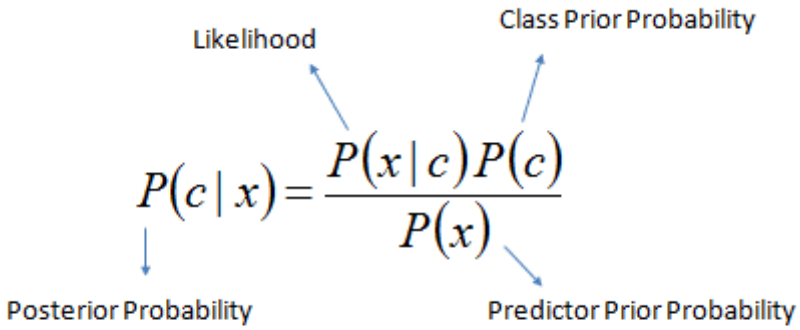
Machine Learning models

Machine learning is the study and the creation of algorithms that learn from given data and examples. It is a particular approach to artificial intelligence.Tom M. Mitchell (an american computer scientist) defines machine learning as "A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P* if its performance at tasks in *T*, as measured by *P*, improves with experience *E*" . In machine learning we have four major models; supervised, semi-supervised,unsupervised and reinforcement.

I. **Supervised learning:** if we have the Input and the Output variable then it is a supervised learning. In this case we only need to map the function between the inputs and the outputs. Supervised learning could be divided into two other sub-categories; Classification and regression: - **Classification:** when the output is a categorical variable - **Regression:** when the output variables are continuous values.

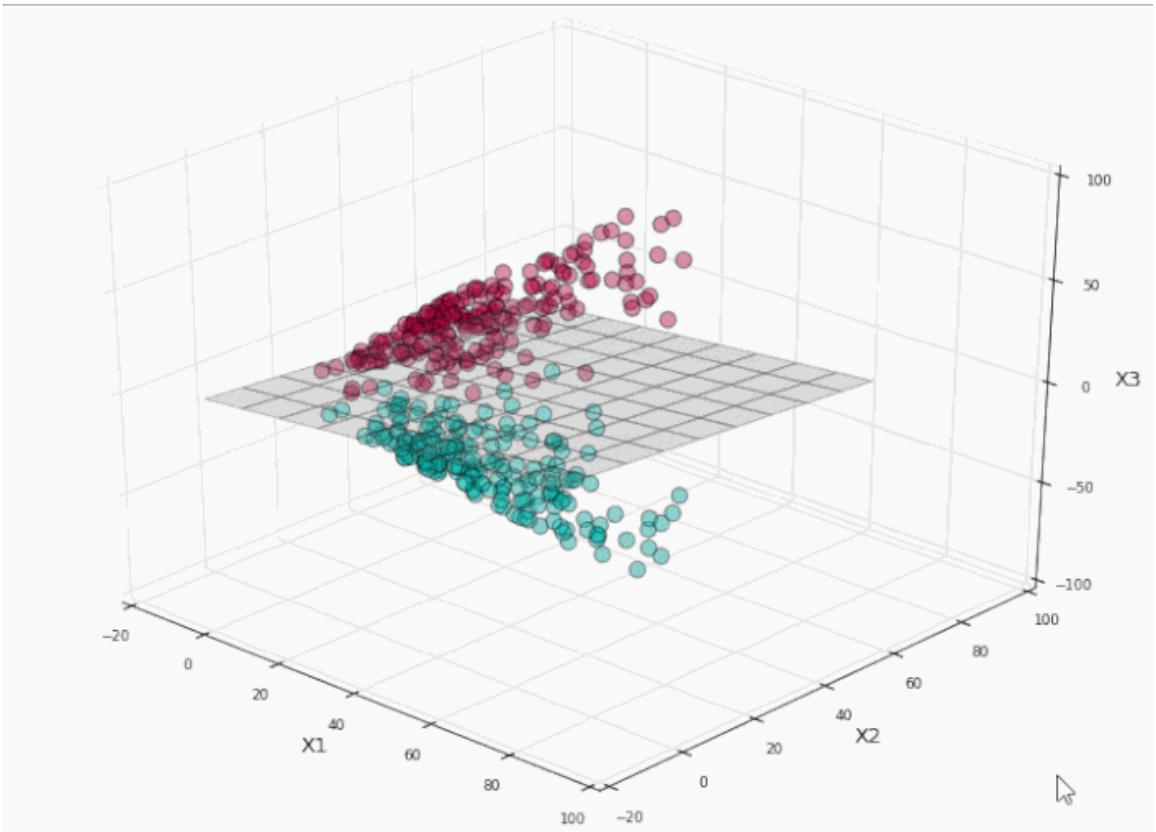
Let's discover some supervised learning algorithms:

- **Naive Bayes:** this classification algorithm is based on the the Bayes' theorem.



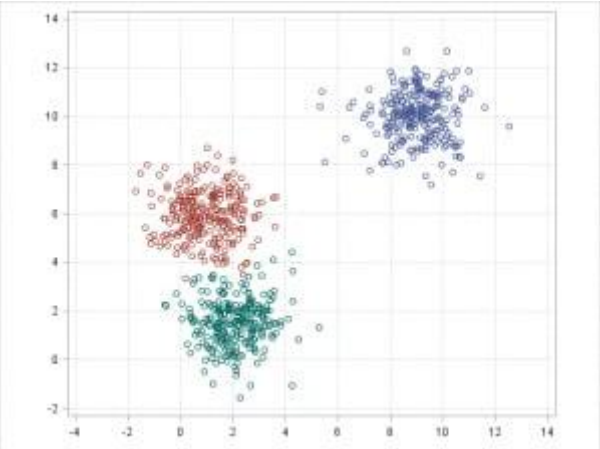
$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

- **Decision Trees:** are machine learning algorithms that predict the possible outputs thanks to a tree-like graph,the entire data is represented as a root node and the final leafs are called Terminal Nodes.Dividable nodes are known as Decision Nodes.
- **Support Vector Machines:** are binary classifiers used to identify a separating hyper-plane of data that are represented in a multi-dimensional space.Thus, that hyper-plane is not necessary a simple line.

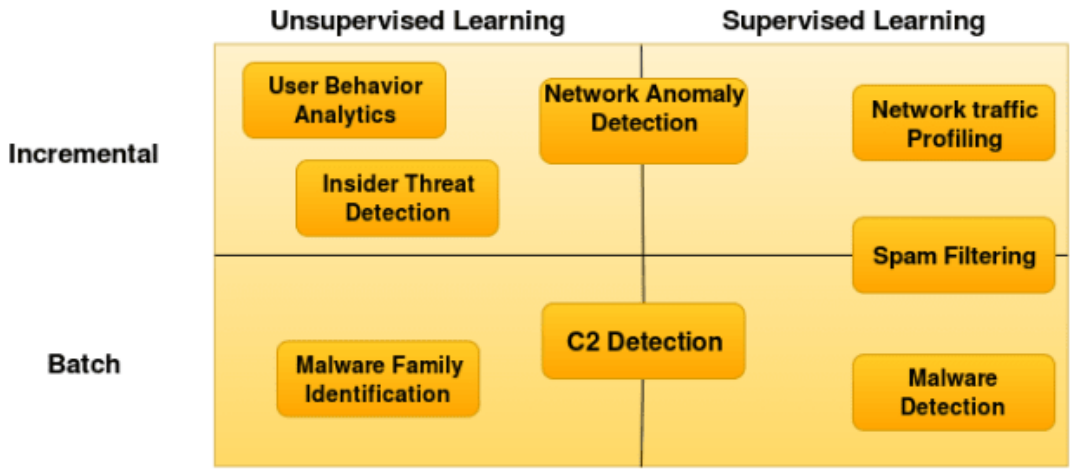


II. **Semi-supervised:** this model is not fully supervised while it contains both labeled and unlabeled data. This model is used generally to improve the learning accuracy. - **Unsupervised:** If we don't have information about the output variables then it is unsupervised learning.The

model is trained totally with unlabeled data.Clustering is one of the most well known unsupervised techniques.



III. **Reinforcement:** in this model the agent is being optimized based on the feedback from the environment (the reward)

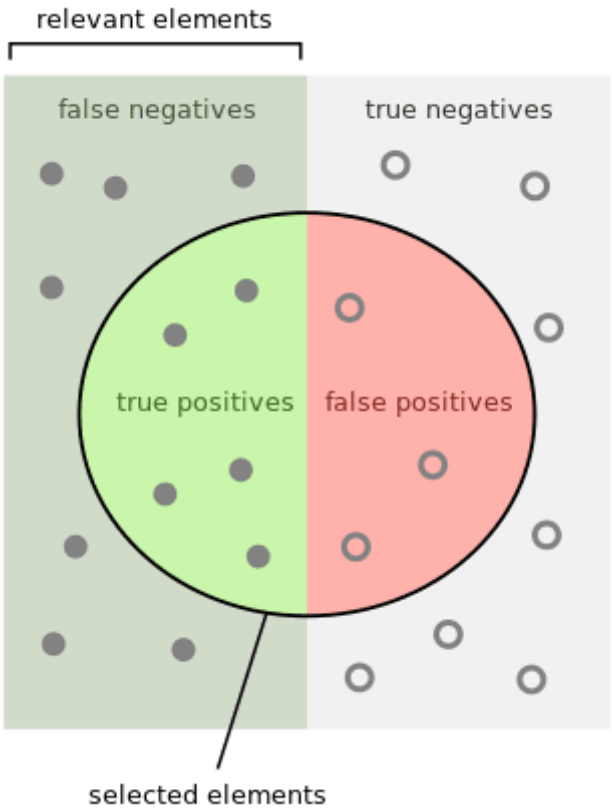


Machine learning steps

In order to build a Machine learning model our project need to follow two major phases; training and experimenting. During the training phase a feature engineering operation is needed because it is critical to feed the machine learning model with a well defined features. Not all the data is useful in our project. After choosing the machine learning algorithm that we are going to use, we feed it by the chosen data. After training, we need to put the model into a test or what we call an experience to evaluate the model based on many evaluation metrics.

Machine learning evaluation metrics

Building a machine learning model is a methodological process. Thus, in order to test our machine learning model performance we need to use a well-defined metrics based on scientific formulas: all these formulas are needing four parameters; false positive, true positive, false negative and true negative.



Notation

- tp = True Positive
- fp = False Positive
- tn = True Negative
- fn = False Negative

Precision

Precision or Positive Predictive Value, is the ratio of the positive samples that are correctly classified by the the total number of positive classified samples.Simply it is the number of the

$$precision = \frac{tp}{tp+fp}$$

found samples were correct hits.

Recall

Recall or True Positive Rate, is the ratio of true positive classifications by the total number of positive samples in the dataset. It represents how many of the true positives were found.

$$Recall = \frac{tp}{tp+fn}$$

F-Score

F-Score of F-Measure, is a measure that combines precision and recall in a one harmonic formula

Accuracy

Accuracy is the ratio of the total correctly classified samples by the total number of samples. This measure is not sufficient by itself,because it is used when we have equal number of classes.

$$F - Score = 2 \times \frac{Precision \times Recall}{Precision \times Recall}$$

Confusion Matrix

Confusion matrix is is a table that is often used to describe the performance of a classification model.

Machine learning python frameworks

As a programming language we used python for many reasons. First comparing to other languages it is more productive and flexible than Java and C++.According to thestateofai.com 78% of developers are using python in their Artificial intelligence projects that means a better documentation and support from the development community. Python is coming with external, easy and advanced machine learning packages in terms of run-time and complexity. The following are some of the most used Python libraries in Machine learning:

• **SciPy** : it is used for mathematics and engineering field in general

• **NumPy** : it is used to manipulate large multi-dimensional arrays and linear algebra

• **MatplotLib** : it provides great data visualization capabilities including: Confusion Matrix,Hitmaps,linear plots

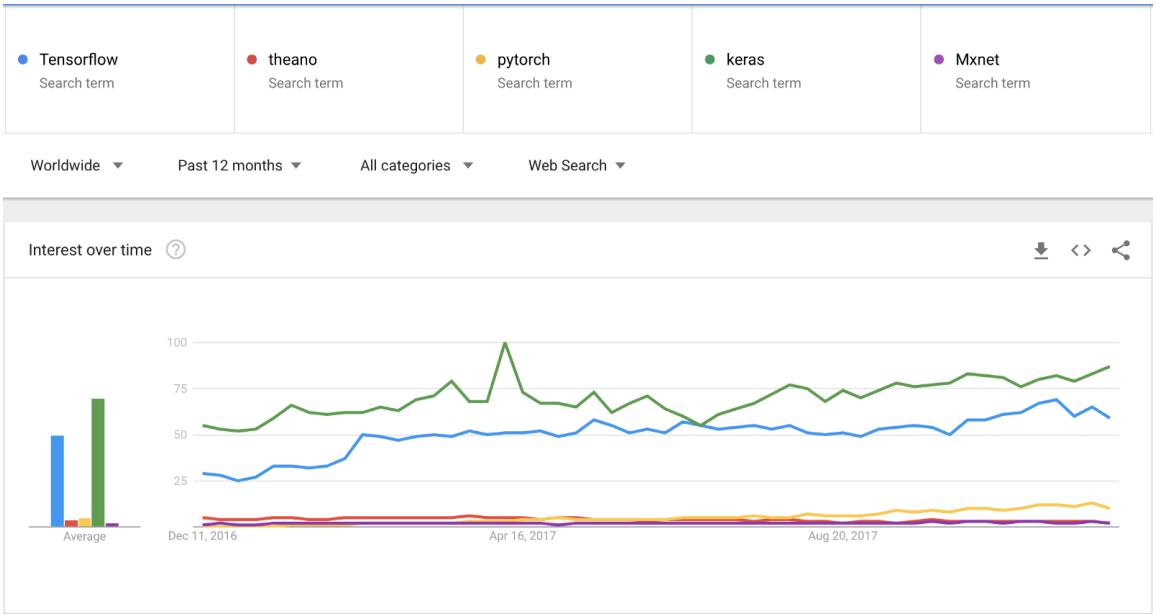
• **Tensorflow** : is an open-source library for machine intelligence and numerical computation developed by Google Brain Team within Google's Machine Intelligence research organization .You can deploy computation to one or more CPUs and GPUs.

• **Keras** : is an open-source neural network library written in Python running on top of TensorFlow to ease the experimentation and the evaluation of the neural networks model.

• **Theano** : is an open source neural network library written in Python running on top of TensorFlow to ease the experimentation and the evaluation of the neural networks model.

To install any Python library this command will do the job : *pip install Package-Here*

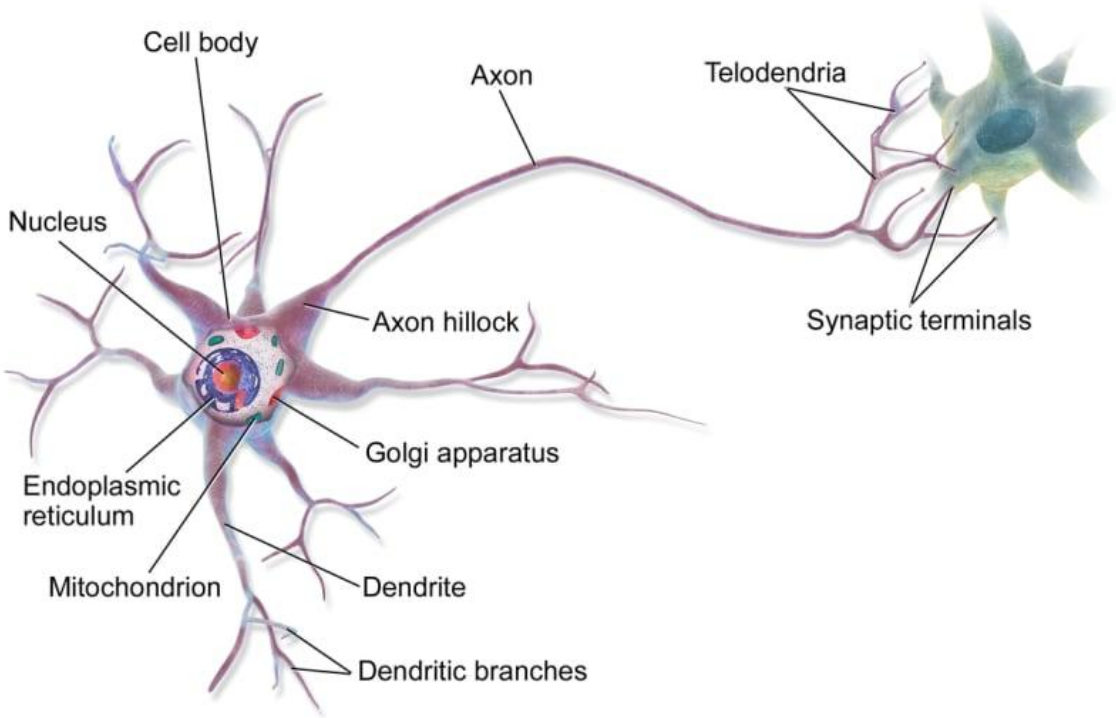
The following graph illustrates a comparison between some machine learning frameworks made by Favio Vázquez especially Deep learning frameworks



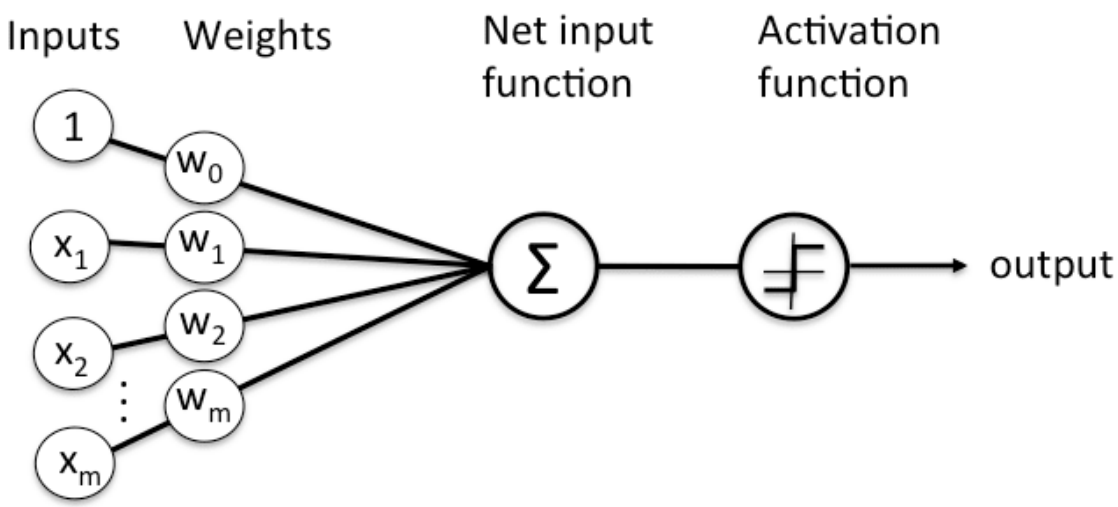
Wait, but what is Deep Learning?

Artificial Neural networks and Deep Learning:

The main goal of Artificial neural networks is to mimic how the brain works.To have a better understanding let's explore how a human brain actually works.Human brain is a fascinated complex entity with many different regions to perform various tasks like listening, seeing, tasting and so on. If the human brain is using many regions to perform multiple tasks so logically every region act using a specific algorithm for example an algorithm for seeing, an algorithm for hearing etc...Right? Wrong! The brain is working using ONE Algorithm. This hypothesis It is called The "one learning algorithm" hypothesis. There is some evidence that the human brain uses essentially *the same algorithm* to understand many different input modalities. For more information check Ferret experiments, in which the "input" for vision was plugged into auditory part of brain, and the auditory cortex learns to "see." The cell that compose the neuron system is called a neuron.The information transmission is happening using electrochemical signalling and propagation is done thanks to the neuron dendrites.



The analogy of the human brain neuron in machine learning is called a perceptron. All the input data is summed and the output applies an activation function. We can see activation functions as information gates.

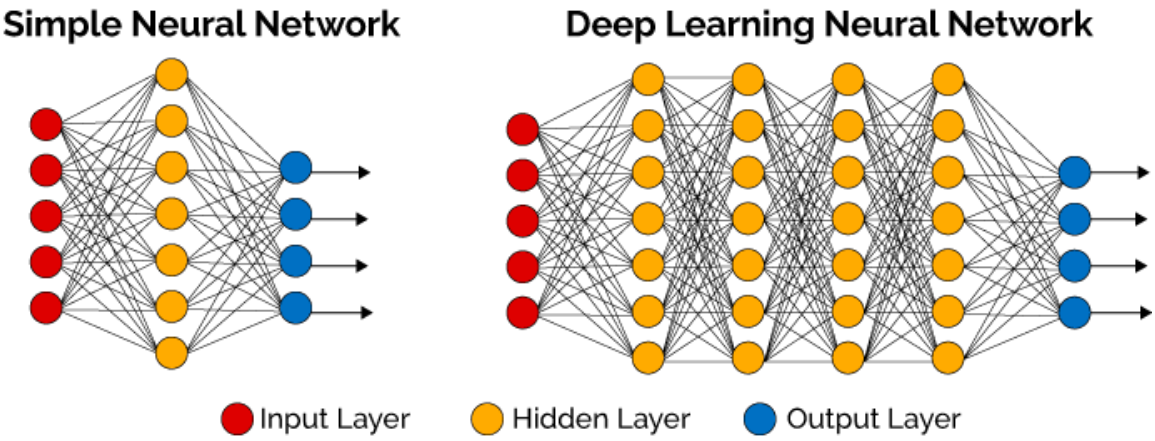


> **PS: " The analogy between a perceptron and a human neuron is not totally correct. It is used just to give a glimpse about how a perceptron works. The human mind is so far more complicated than Artificial neural networks. There are few similarities but a comparison between the mind and Neural networks is not really correct."**

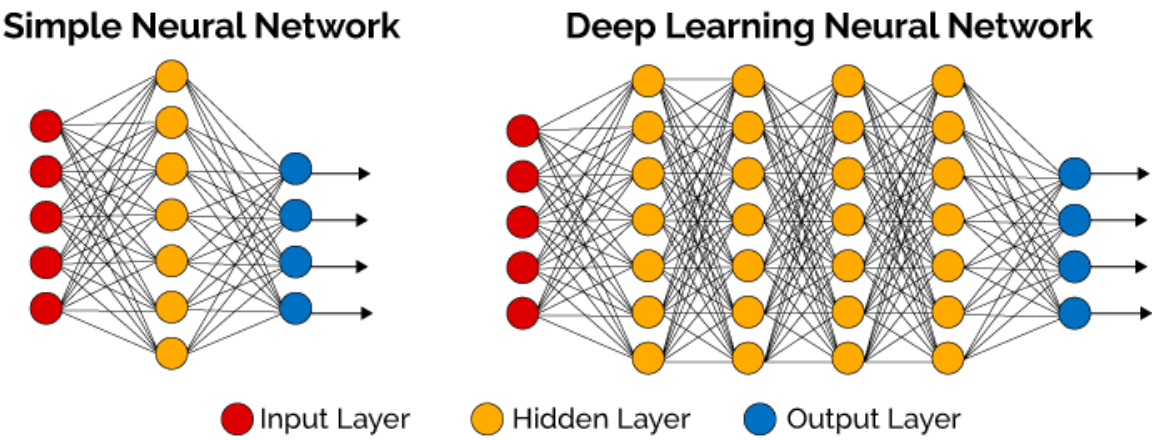
There are many used activation functions:

- **Step Function** : Every output node have a predefined threshold value

- **Sigmoid Function** : Sigmoid functions are one of the most widely used activation functions
- **Tanh Function** : Another activation function used is the Tanh function
- **ReLu Function** : It is also called a rectified linear unit.It gives an output x if x is positive and 0 otherwise.



Many connected perceptrons build a simple neural network that consists of three parts: Input layer,hidden layer and an output layer.The hidden layer is playing the inter-communication role in the neural network or sometimes what what we call a Multi-layer perceptron network. If we have more than 3 hidden layers then we are talking about Deep Learning and Deep learning Networks.



According to the data scientist and deep learning experts like the machine learning practitioner Dr. Jason Brownlee; every deep learning model must go thru five steps:

• **Network Definition:** in this phase we need to define the layers.Thanks to Keras this step is easy because it defines neural networks as sequences and to define layers we just need to create a sequence instance with mentioning the number of outputs

• **Network Compiling:** Now we need to compile the network including choosing the optimizing technique like Stochastic Gradient Descent (sgd) and a Loss function (Loss function is used to measure the degree of fit) to evaluate the model we can use Mean Squared Error (mse)

• **Network Fitting:** a Back-Propagation algorithm is used during this step based on the parameters specified in the compiling step.

• **Network Evaluation :** After fitting the network an evaluation operation is needed to evaluate the performance of the model

• **Prediction:** Finally after training the deep learningmodel we now can use it to predict a new malware sample using a testingdataset

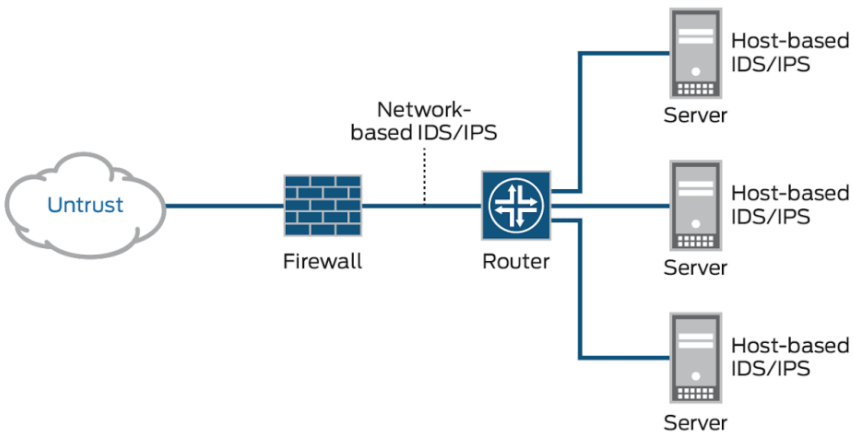
Intrusion detection systems with Machine learning

Dangerous hackers are inventing new techniques in a daily basis to bypass security layers and avoid detection.Thus it is time to figure out new techniques to defend against cyber threats. Intrusion detection systems are a set of devices or pieces of software that play a huge role in modern organizations to defend against intrusions and malicious activities.We have two major intrusion detection system categories:

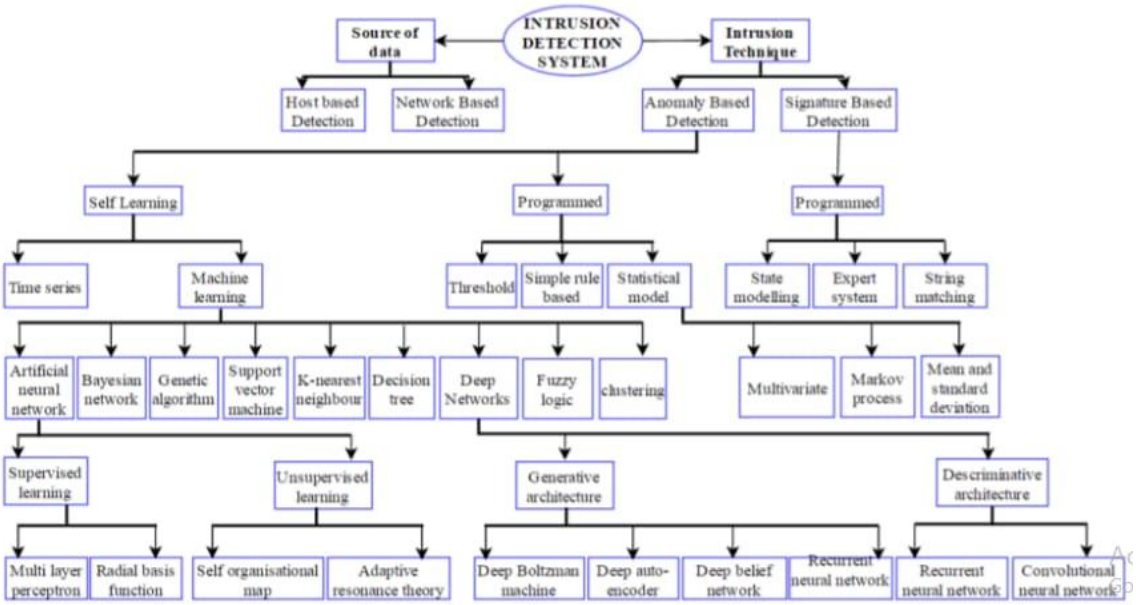
- **Host Based Intrusion Detection Systems (HIDS):** they run on the enterprise hosts to
- **Network Based Intrusion Detection Systems (NIDS):** their role is to detect network anomalies by monitoring the inbound and outbound traffic.

The detection can be done using two intrusion detection techniques:

- **Signature based detection technique:** the traffic is compared against a database of signatures of known threats
- Anomaly-based intrusion technique: inspects the traffic based on the behavior of activities.



Modern organization are facing thousands of threats in a daily basis.That is way the classic techniques could not be a wise solution to defend against them.Many researchers and information security professionals are coming with new concepts,prototypes or models to try solving this serious security issues.For example this is graph shows the different intrusion detection techniques including the discussed machine learning algorithms

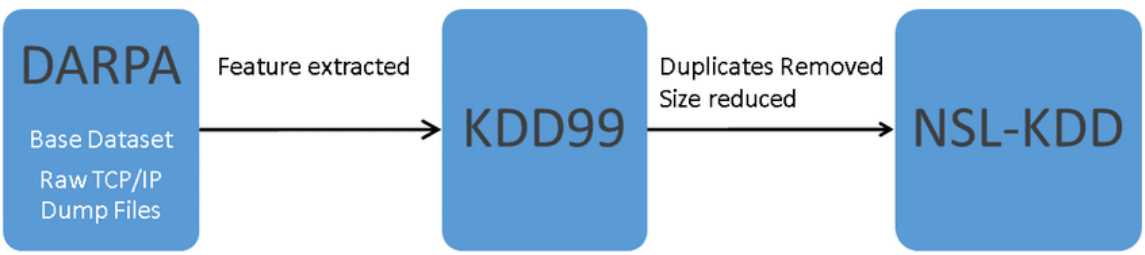


By now, after reading the previous sections we are able to build a Machine learning detection system. As discussed before the first step is Data processing.The are many publicly available datasets in the wild used by data scientist to train machine learning models.You can download some of them from here:

- The ADFA Intrusion Detection Datasets: https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-IDS-Datasets/
- Publicly available pcap files: http://www.netresec.com/?page=PcapFiles
- The Cyber Research Center - DataSets: https://www.westpoint.edu/crc/SitePages/DataSets.aspx
- The NSL-KDD dataset: https://github.com/defcom17/NSL_KDD

The NSL-KDD is one of the most used datasets in intrusion detection anomaly based models.It contains different attacks categories: DoS, Probe, U2R and R2L.

It is an enhanced dataset from the KDD99 dataset



After choosing the feature that you are going to work on and splitting the dataset into two sub-datasets for the training and the experience (They should not be the same) you can choose one of the machine learning algorithms represented in the graph of intrusion detection techniques and train your model.Finally when you finish the training phase it is time to put your model to the test and check its accuracy based on the machine learning evaluation metrics. To explore

some of the tested models i recommend taking an eye on "Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey" research paper.

There are a lot of talks about the promise of machine learning or AI ininformation security but in the other side there is a debate and some concerns about it. To discover more about Machine learning promises in cyber security it is highly recommended to watch **Thomas Dullien** Talk : " **Machine Learning, offense, and the future of automation**" from here:

You can also download the presentation slides from this link: Presentation Slides

Summary

This article is a fair overview of machine learning in information security.We discussed the required fundamentals in every machine learning project starting from the fundamentals to gaining the skills to build a machine learning projects.We took intrusion detection systems as real world case study.

References

1. https://www.slideshare.net/idsecconf/jim-geovedi-machine-learning-for-cybersecurity
2. https://blog.capterra.com/artificial-intelligence-in-cybersecurity/

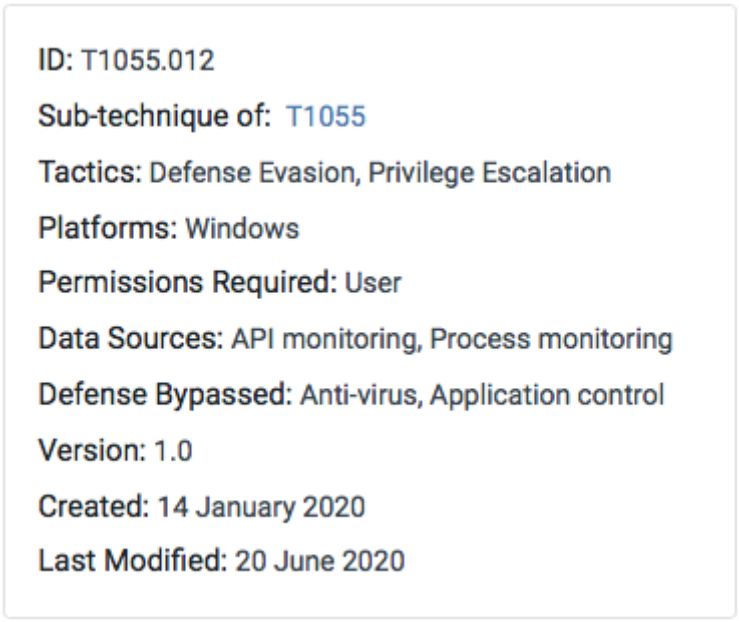
Azure Sentinel: Process Hollowing (T1055.012) Analysis

In this article, we are going to explore a technique called Process Hollowing.

Before jumping into the detection part, it is essential to explore some important terminologies.

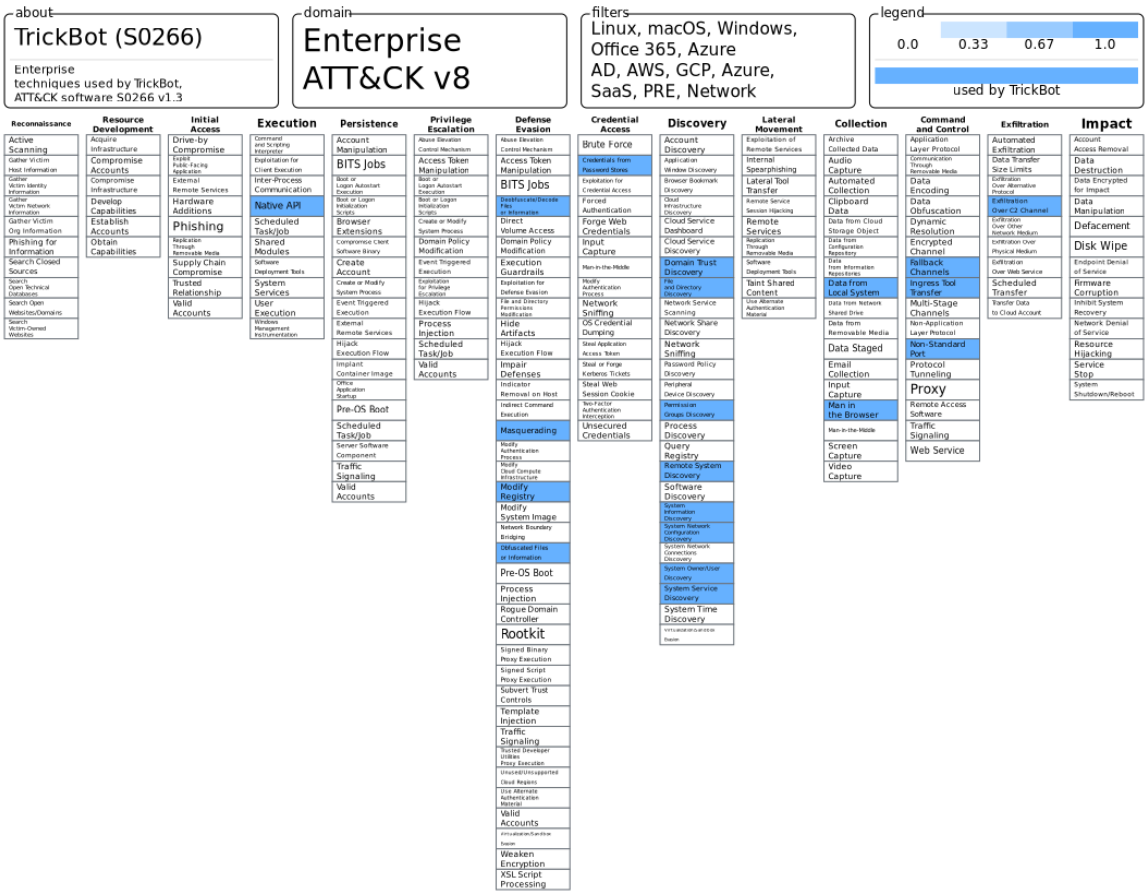
According to MITRE:

"Process hollowing (T1055.012) is commonly performed by creating a process in a suspended state then unmapping/hollowing its memory, which can then be replaced with malicious code. A victim process can be created with native Windows API calls such as CreateProcess, which includes a flag to suspend the processes primary thread. At this point the process can be unmapped using APIs calls such as ZwUnmapViewOfSection or NtUnmapViewOfSection before being written to, realigned to the injected code, and resumed via VirtualAllocEx, WriteProcessMemory, SetThreadContext, then ResumeThread respectively"



To learn more about Process hollowing, i highly recommend you to check this piece from Elastic: https://www.elastic.co/blog/ten-process-injection-techniques-technical-survey-common-and-trending-process

This technique is widely used by adversaries such as Duqu and TrickBot



The following pieces by Jonathan Johnson and David Polojac from Specterops deep dive into the detection engineering aspects of process hollowing

- Engineering Process Injection Detections - Part 1: Research
- Engineering Process Injection Detections — Part 2: Data Modeling
- Engineering Process Injection Detections — Part 3: Analytic Logic

For the detection we are going to use Azure Sentinel and sysmon. Sysmon can be downloaded from here:

https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon

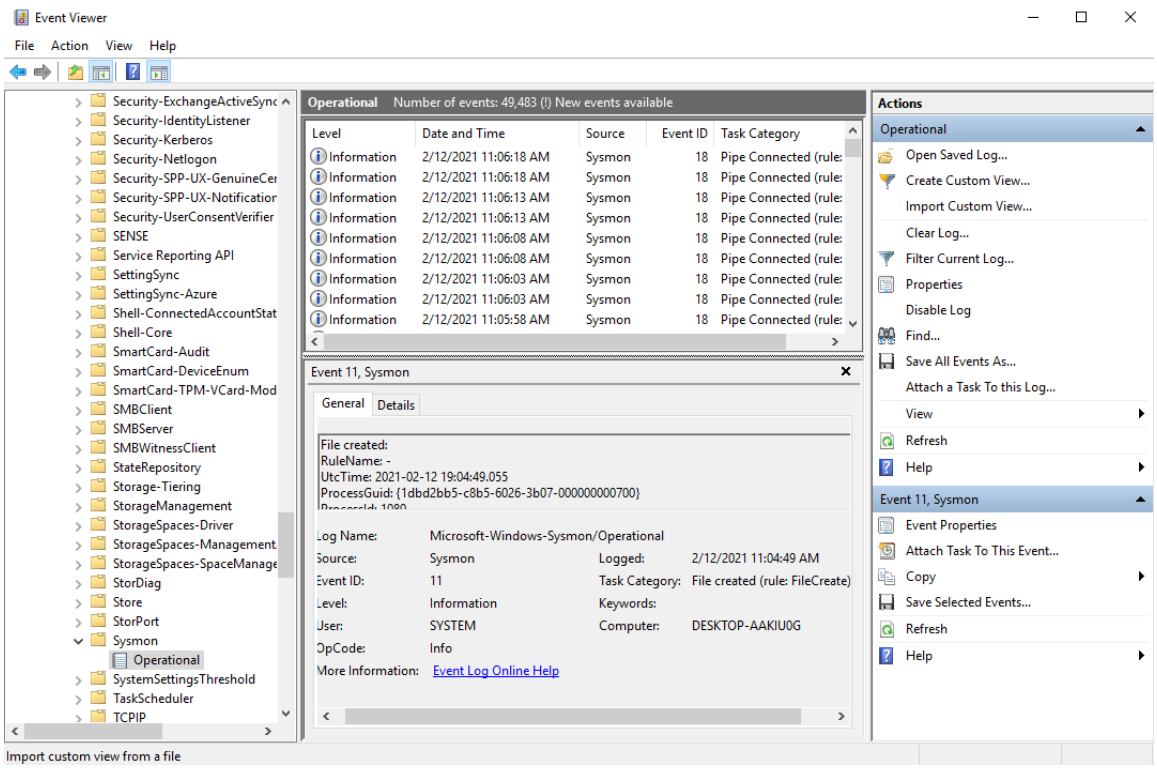
To install it, run the following command as an administrator:

```
sysmon.exe -accepteula -i <CONFIG_FILE_HERE>
```

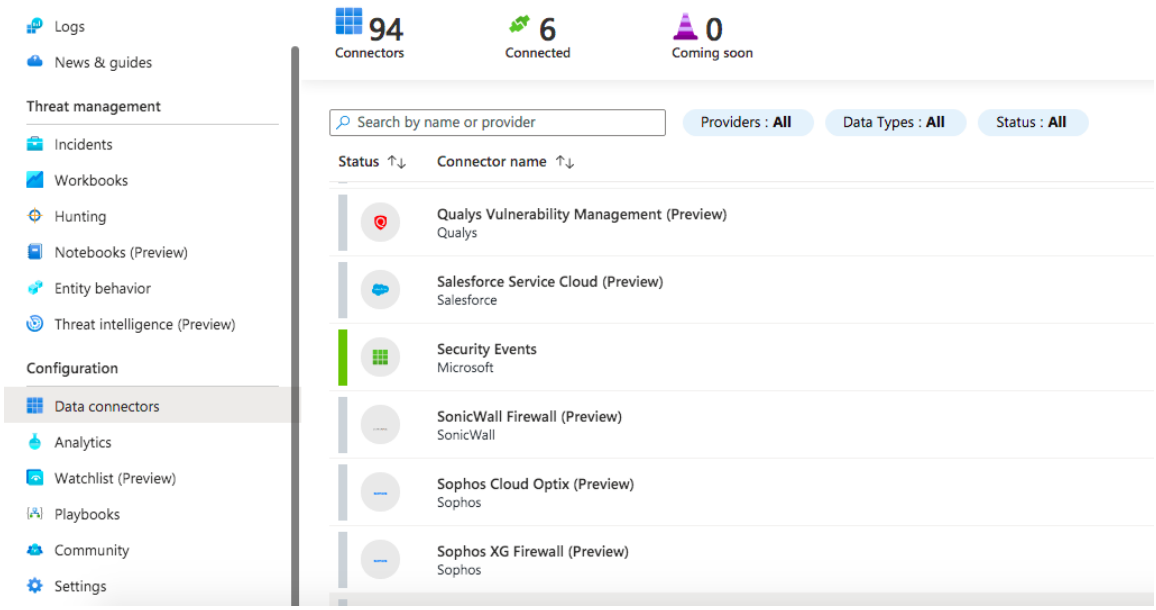
You can use the following config file by ION-STORM:

https://github.com/ion-storm/sysmon-config

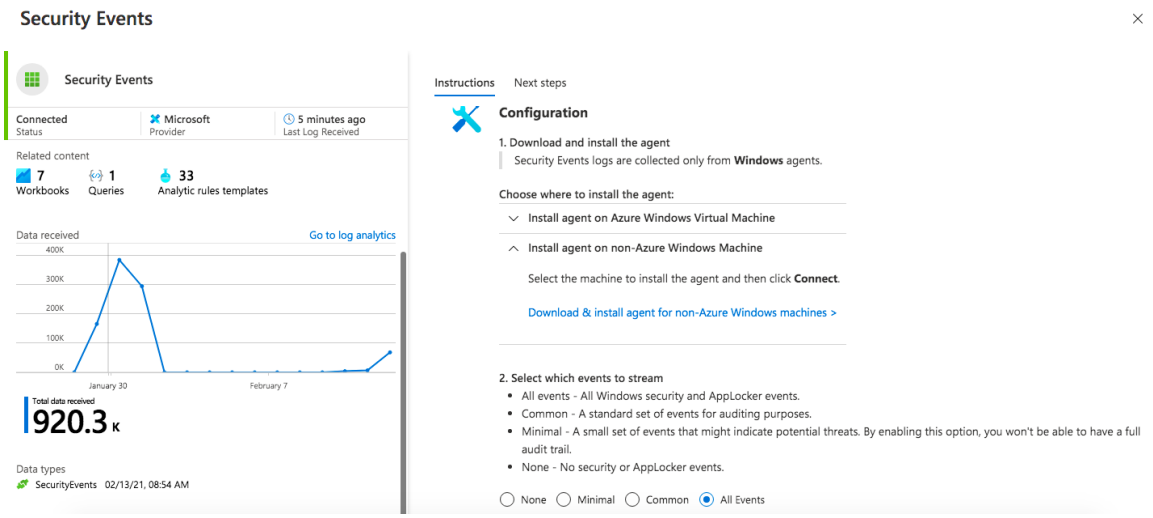
To explore sysmon events, use Windows Event Viewer: Applications and services logs -\> Microsoft -\> Windows -\> Sysmon -\> Operational



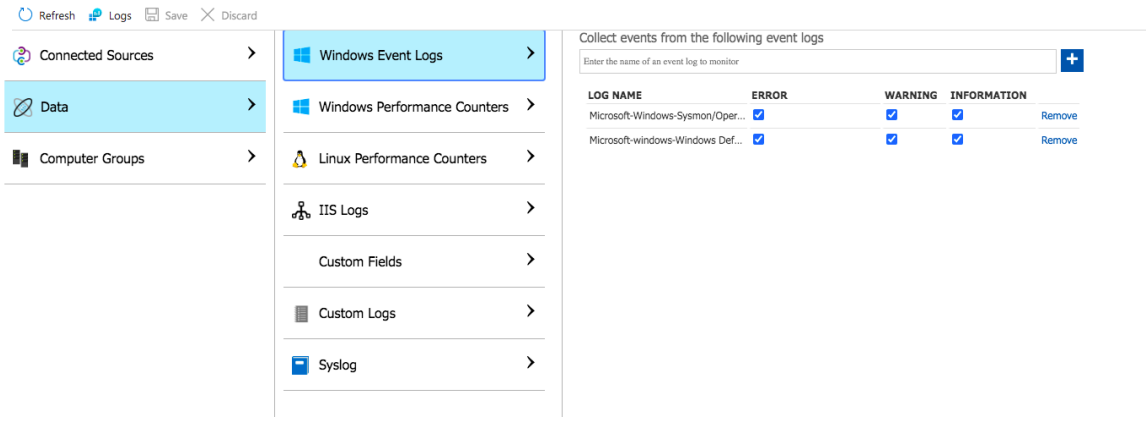
To send sysmon events to Azure sentinel, deploy a new connector (Security Events) to start with Windows Event logs



Install the agent.

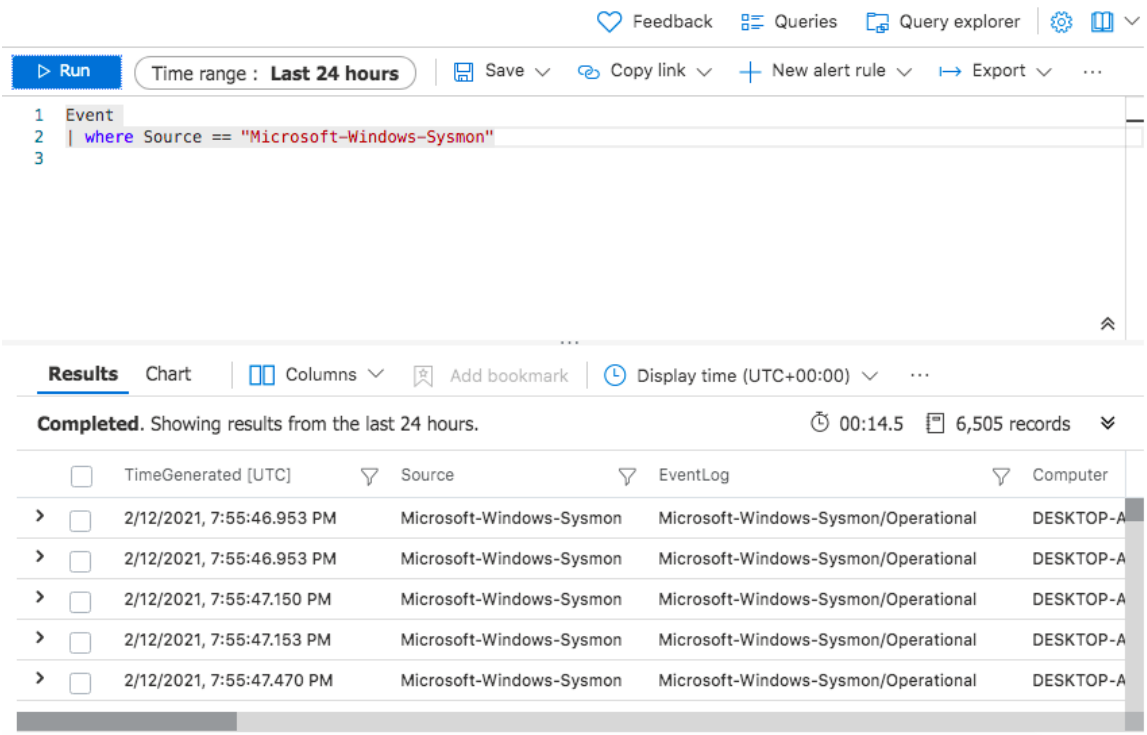


Now go to Settings -\> Workspace Settings -\> Advanced settings -\> Data -\> Windows Event Logs and add the following event log name: **Microsoft-Windows-Sysmon/Operational**

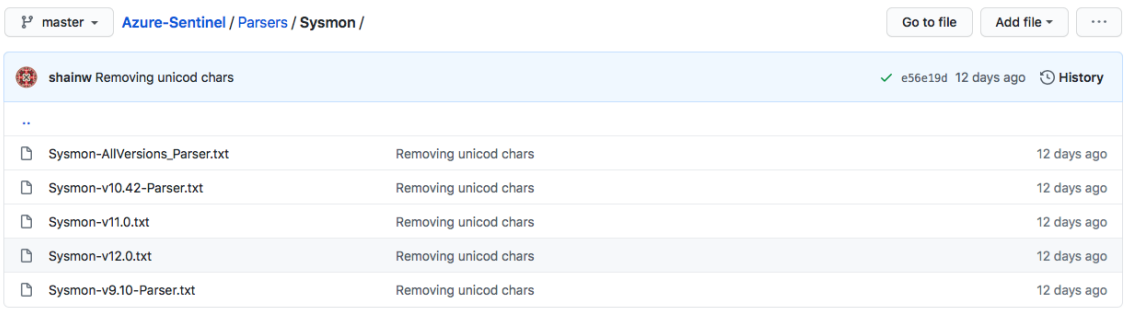


To check the events go to Azure Sentinel **Logs** section and run the following query:

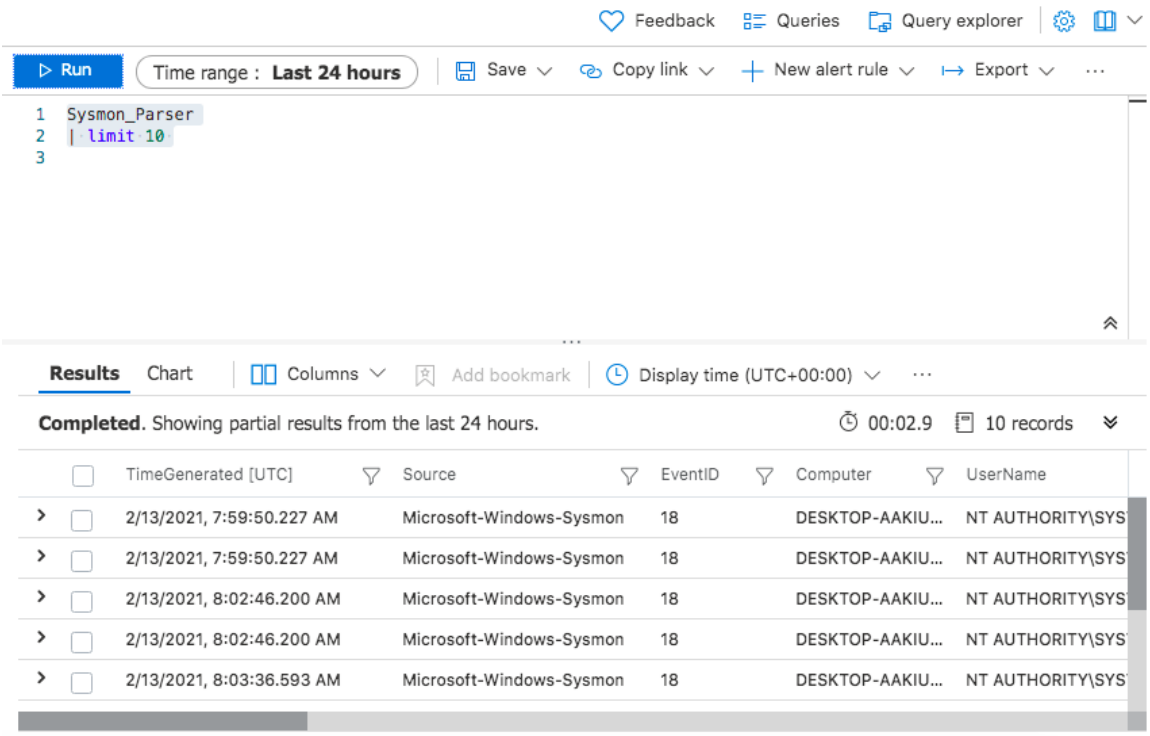
```
Event
| where Source == "Microsoft-Windows-Sysmon"
```



As you will notice the EventData fields are not parsed and filtered. Thus, it is recommended to use one of Azure Sentinel sysmon parsers: https://github.com/Azure/Azure-Sentinel/tree/master/Parsers/Sysmon

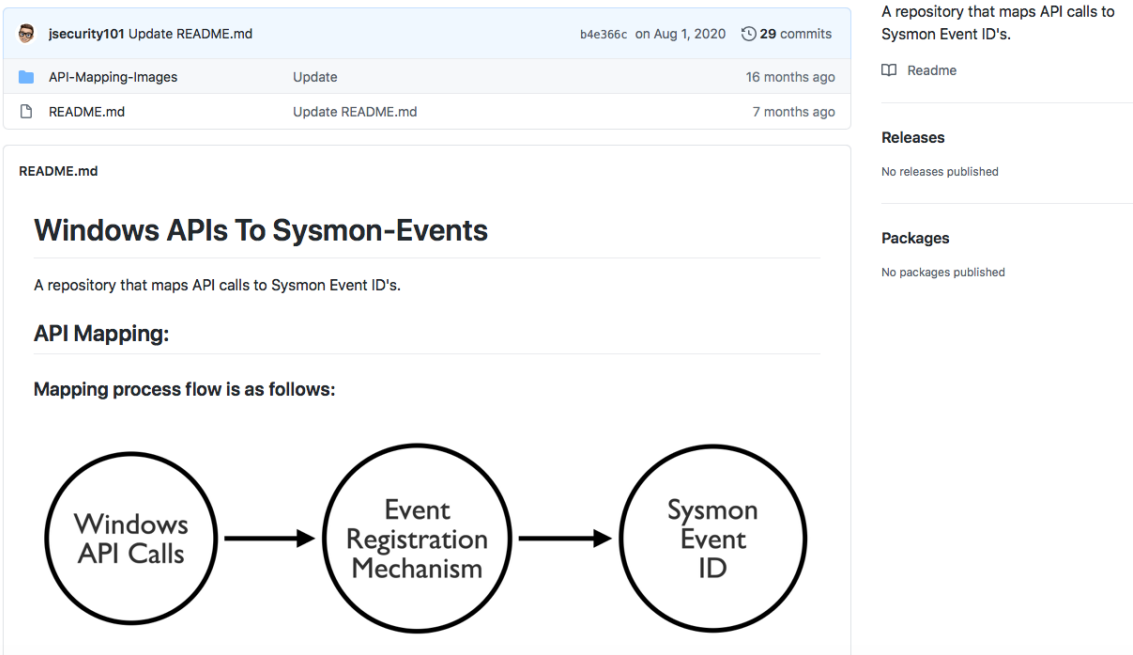


To use the parser, copy the file content in log analytics and save it as a function (e.g Sysmon_Parser). Now the events are well parsed:



To correlate APIs with Events, a mapping phase is needed for a better visibility. Thankfully, you can use these sheets:

- https://github.com/hunters-forge/API-To-Event
- https://github.com/jsecurity101/Windows-API-To-Sysmon-Events

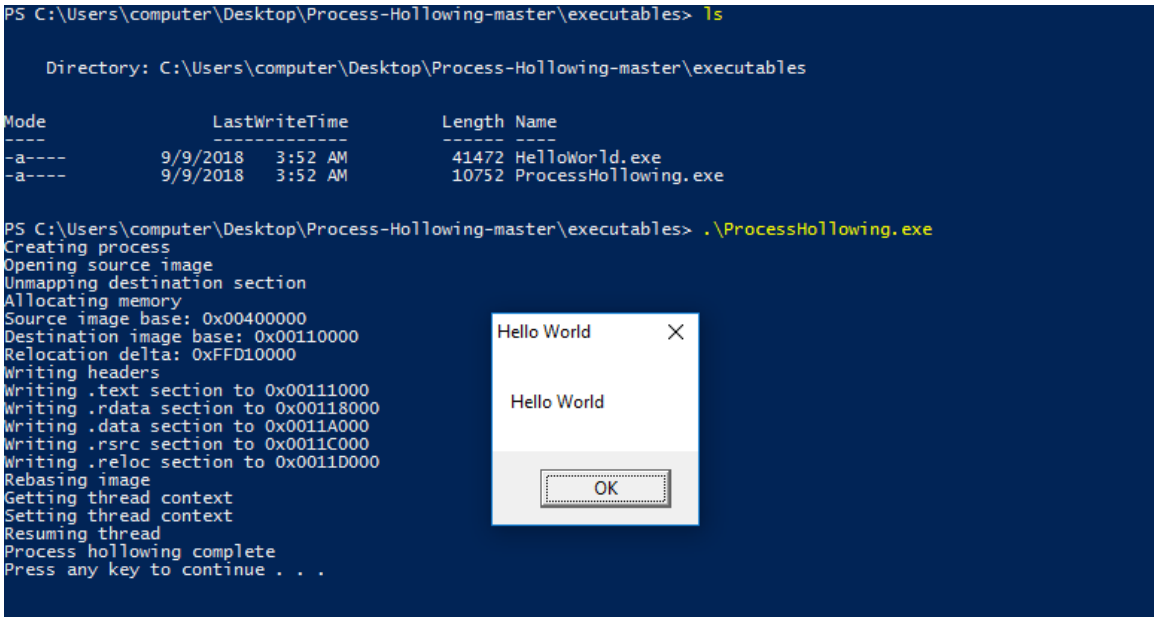


More details about mapping can be found here: Uncovering The Unknowns

Now we know what sysmon EventIDs to watch

	A	B	C	D
1	**API**	**Event Registration Mechanism**	**Data Sensor**	**Event ID**
2	CreateProcess	PsSetCreateProcessNotifyRoutine	Sysmon	1
3	CreateProcessAsUser	PsSetCreateProcessNotifyRoutine	Sysmon	1
4	CreateProcessWithToken	PsSetCreateProcessNotifyRoutine	Sysmon	1
5	CreateProcessWithLogon	PsSetCreateProcessNotifyRoutine	Sysmon	1
6	CreateProcessInternal	PsSetCreateProcessNotifyRoutine	Sysmon	1

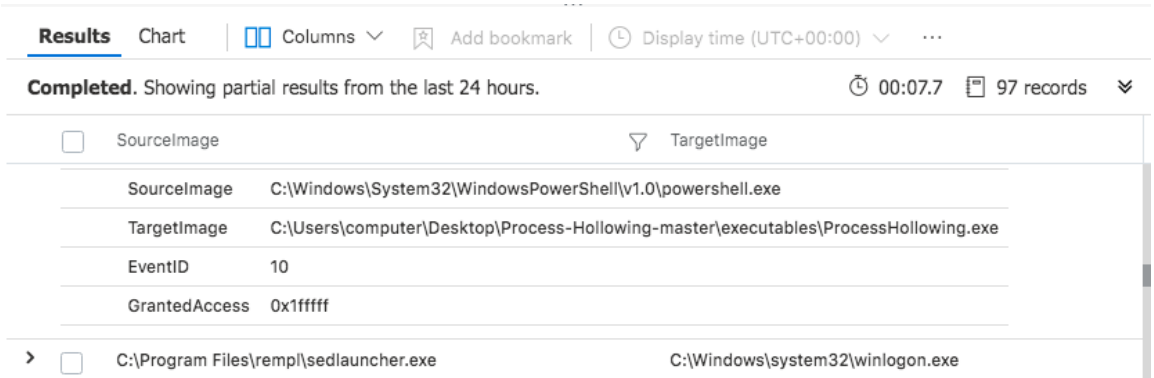
Let's perform a process hollowing technique using the following poc:
https://github.com/m0n0ph1/Process-Hollowing



Go to Azure Sentinel logs console

```
Sysmon_Parser
| where EventID in ("1","10")
| project SourceImage, TargetImage, EventID, GrantedAccess
```

- EventID 1: Process Created
- EventID 10: Process Accessed
- The project operator: Only the columns specified in the arguments are included in the result.



In our case, the access rights used by the POC is 0x1fffff which is PROCESS_ALL_ACCESS even though according to Jonathan Johnson's research process hollowing only needs the following rights:

PROCESS_VM_WRITE

PROCESS_VM_OPERATION

PROCESS_SUSPEND_RESUME

PROCESS_CREATE_PROCESS

Module 23 - Azure Sentinel - Send Events with Filebeat and Logstash

Filebeat Logstash to Azure Sentinel

In this new post we are going to explore how to send events/logs to Azure Sentinel using Filebeat and Logstash.

How to install and Configure Filebeat:

Filebeat can be downloaded from here: https://www.elastic.co/downloads/beats/filebeat

To install filebeat run the following commands (on Ubuntu 18 in my case)

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key
add -
```

```
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo
tee -a /etc/apt/sources.list.d/elastic-7.x.list
```

```
sudo apt update
```

```
sudo apt install filebeat
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  grub-pc-bin linux-azure-5.3-cloud-tools-5.3.0-1020 linux-azure-5.3-cloud-tools-5.3.0-1022
  linux-azure-5.3-cloud-tools-5.3.0-1028 linux-azure-5.3-cloud-tools-5.3.0-1031
  linux-azure-5.3-cloud-tools-5.3.0-1032 linux-azure-5.3-cloud-tools-5.3.0-1034
  linux-azure-5.3-cloud-tools-5.3.0-1035 linux-azure-5.3-headers-5.3.0-1020 linux-azure-5.3-headers-5.3.0-1022
  linux-azure-5.3-headers-5.3.0-1028 linux-azure-5.3-headers-5.3.0-1031 linux-azure-5.3-headers-5.3.0-1032
  linux-azure-5.3-headers-5.3.0-1034 linux-azure-5.3-headers-5.3.0-1035 linux-azure-5.3-tools-5.3.0-1020
  linux-azure-5.3-tools-5.3.0-1022 linux-azure-5.3-tools-5.3.0-1028 linux-azure-5.3-tools-5.3.0-1031
  linux-azure-5.3-tools-5.3.0-1032 linux-azure-5.3-tools-5.3.0-1034 linux-azure-5.3-tools-5.3.0-1035
  linux-azure-5.4-cloud-tools-5.4.0-1023 linux-azure-5.4-cloud-tools-5.4.0-1025
  linux-azure-5.4-cloud-tools-5.4.0-1026 linux-azure-5.4-cloud-tools-5.4.0-1031
  linux-azure-5.4-cloud-tools-5.4.0-1034 linux-azure-5.4-headers-5.4.0-1023 linux-azure-5.4-headers-5.4.0-1025
  linux-azure-5.4-headers-5.4.0-1026 linux-azure-5.4-headers-5.4.0-1031 linux-azure-5.4-headers-5.4.0-1034
  linux-azure-5.4-tools-5.4.0-1023 linux-azure-5.4-tools-5.4.0-1025 linux-azure-5.4-tools-5.4.0-1026
  linux-azure-5.4-tools-5.4.0-1031 linux-azure-5.4-tools-5.4.0-1034
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  filebeat
0 upgraded, 1 newly installed, 0 to remove and 69 not upgraded.
Need to get 9326 kB of archives.
After this operation, 30.5 MB of additional disk space will be used.
```

Filebeat comes with some available log modules such as the following modules

```
Enabled:

Disabled:
activemq
apache
auditd
aws
azure
barracuda
bluecoat
cef
checkpoint
cisco
coredns
crowdstrike
cyberark
cylance
elasticsearch
envoyproxy
f5
fortinet
gcp
google_workspace
```

For example, let's enable the system module:

```
sudo filebeat modules enable system
```

Edit the config file:

```
sudo vi /etc/filebeat/filebeat.yml
```

```
###################### Filebeat Configuration Example #########################

# This file is an example configuration file highlighting only the most common
# options. The filebeat.full.yml file from the same directory contains all the
# supported options with more comments. You can use it as a reference.
#
# You can find the full configuration reference here:
# https://www.elastic.co/guide/en/beats/filebeat/index.html

#========================== Filebeat prospectors =============================

filebeat.prospectors:

# Each - is a prospector. Most options can be set at the prospector level, so
# you can use different prospectors for various configurations.
# Below are the prospector specific configurations.

- input_type: log

  # Paths that should be crawled and fetched. Glob based paths.
  paths:
    - /var/log/*.log
    #- c:\programdata\elasticsearch\logs\*
```

Comment Elasticsearch Output section and uncomment Logstash output:

```
#----------------------------- Elasticsearch output --------------------------------
#output.elasticsearch:
  # Array of hosts to connect to.
  # hosts: ["localhost:9200"]

  # Optional protocol and basic auth credentials.
  #protocol: "https"
  #username: "elastic"
  #password: "changeme"

#----------------------------- Logstash output --------------------------------
output.logstash:
  # The Logstash hosts
  hosts: ["localhost:5044"]

  # Optional SSL. By default is off.
  # List of root certificates for HTTPS server verifications
  #ssl.certificate_authorities: ["/etc/pki/root/ca.pem"]
```

Start Filebeat

```
sudo service filebeat start
```

To check its status type:

```
sudo service filebeat status
```

How to install and Configure Logstash

Logstash is a free and open server-side data processing pipeline that ingests data from a multitude of sources, transforms it, and then sends it to your favorite "stash." (Source: Elastic.io)

Logstash can be downloaded from here: https://www.elastic.co/downloads/logstash

```
sudo apt install -y openjdk-8-jdk
```

```
sudo apt-get install logstash
```

Enter /etc/logstash/conf.d/

```
cd /etc/logstash/conf.d/
```

Create a new config file:

```
sudo vi Azure-Sentinel.conf
```

add the folllowing content

```
input {
      beats {
          port => "5044"
      }
  }
  filter {
  }
  output {
      microsoft-logstash-output-azure-loganalytics {
        workspace_id => "<your workspace id>"
        workspace_key => "<your workspace key>"
        custom_log_table_name => "tableName"
      }
  }
```

More configurations can be found here: https://docs.microsoft.com/en-us/azure/sentinel/connect-logstash

Start logstash

```
sudo service logstash start
```

Now you can query events by selecting the table name

Azure Sentinel: Using Custom Logs and DNSTwist to Monitor Malicious Similar Domains

In this article, we are going to explore how to monitor similar domains to yours, in order to protect your users from being victims of social engineering attacks.

When performing computer-based social engineering attacks such as phishing, attackers buy similar domains to yours in order to trick your users. This is why keeping an eye on similar domains is essential to avoid such attacks.

First we need to find these domains. One of the tools that helps you to generate similar domains is "DNS Twist". You can find it here: https://github.com/elceef/dnstwist

According to DNS Twist developers:

> "DNS fuzzing is an automated workflow for discovering potentially malicious domains targeting your organisation. This tool works by generating a large list of permutations based on a domain name you provide and then checking if any of those permutations are in use. Additionally, it can generate fuzzy hashes of the web pages to see if they are part of an ongoing phishing attack or brand impersonation, and much more!"

You can even try to generate some domains online here: https://dnstwist.it

In this demonstration, we are going to use python on Windows to generate similar domains:

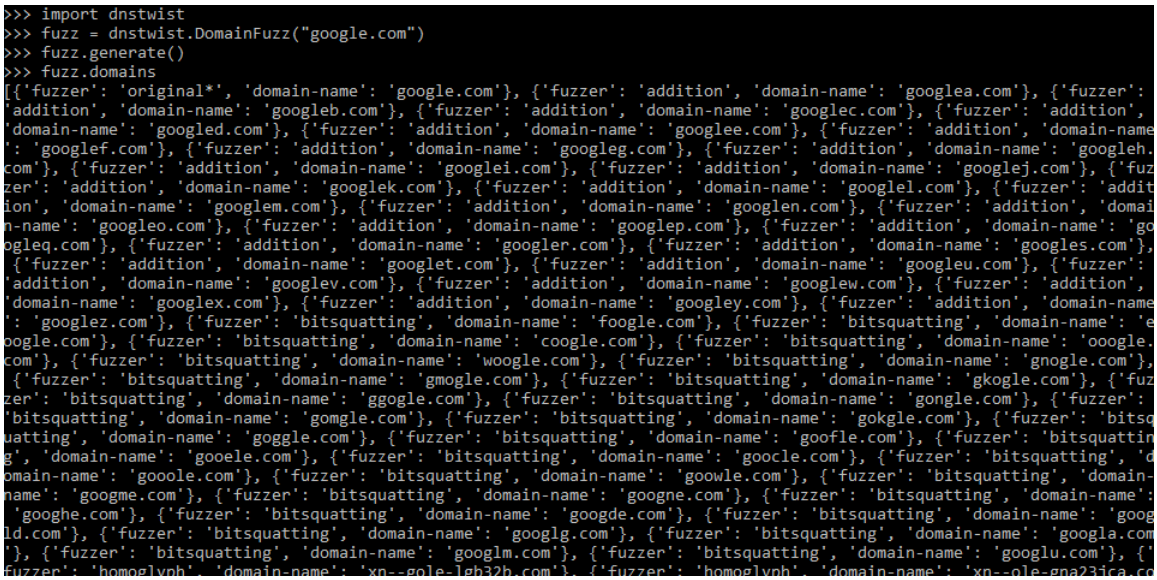
Type the following command to install the python module:

```
py -m pip install dnstwist
```

```
C:\Users\computer>py -m pip install dnstwist
Collecting dnstwist
  Downloading dnstwist-20201228-py3-none-any.whl (10 kB)
Installing collected packages: dnstwist
  WARNING: The script dnstwist.exe is installed in 'C:\Users\computer\AppData\Local\Programs\Python\Python39\Scripts' wh
ich is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed dnstwist-20201228
WARNING: You are using pip version 20.2.4; however, version 21.0.1 is available.
You should consider upgrading via the 'C:\Users\computer\AppData\Local\Programs\Python\Python39\python.exe -m pip instal
l --upgrade pip' command.
```

To generate similar domains, open python terminal and type:

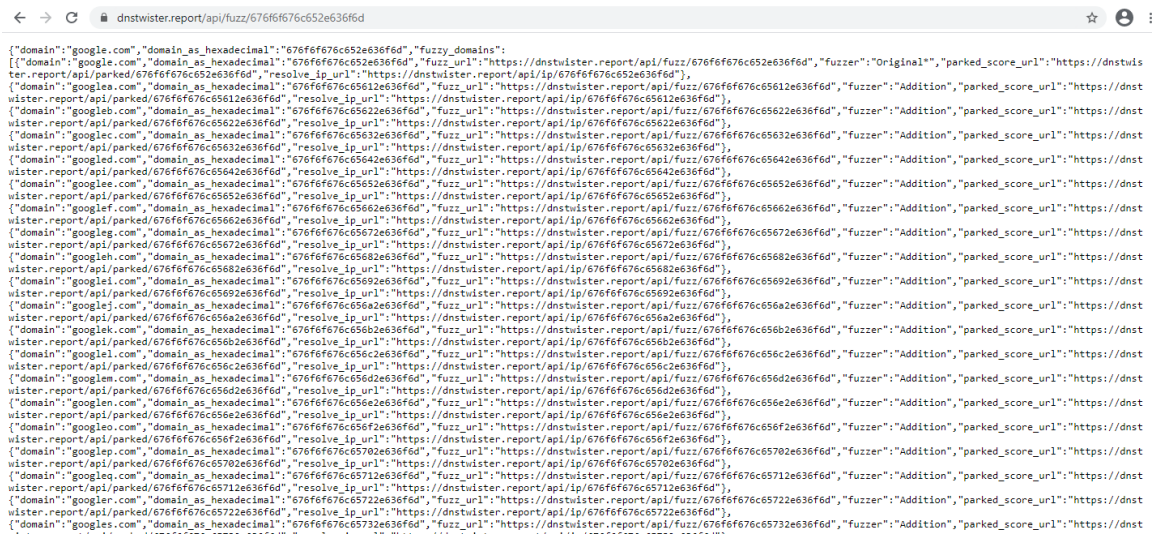
```
import dnstwist
fuzz = dnstwist.DomainFuzz("<YOUR DOMAIN HERE>")
fuzz.generate()
fuzz.domains
```



For example these are some similar domains to "google.com" after parsing only the domain names:

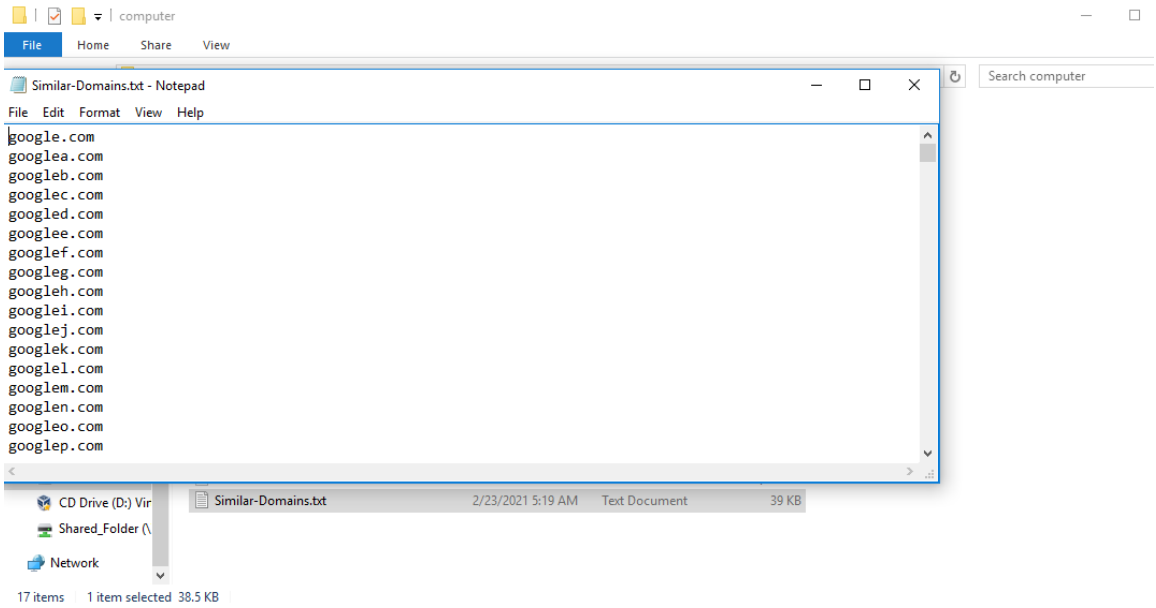
```
>>> for i in range(len(fuzz.domains)):
...     print(fuzz.domains[i]['domain-name'])
...
google.com
googlea.com
googleb.com
googlec.com
googled.com
googlee.com
googlef.com
googleg.com
googleh.com
googlei.com
googlej.com
googlek.com
googlel.com
googlem.com
googlen.com
googleo.com
googlep.com
googleq.com
googler.com
googles.com
googlet.com
googleu.com
googlev.com
googlew.com
googlex.com
googley.com
googlez.com
```

You can also use this API: https://dnstwister.report/api/



To store the similar domains you can build a small script to achieve that. For example the following snippet stores similar domains in a file called "Similar-Domains.txt"

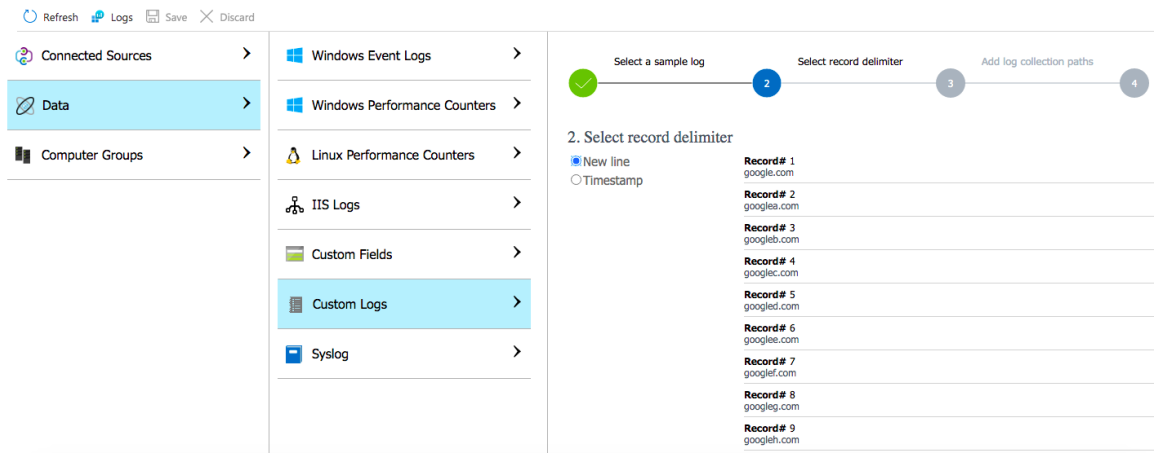
```
>>> with open("Similar-Domains.txt","w") as f:
...     for domain in Domains:
...             f.write('%s\n'  % domain)
...
```



Once, we have a file that contains the similar domains, now we need to send them to sentinel so later we can create rules based on them.

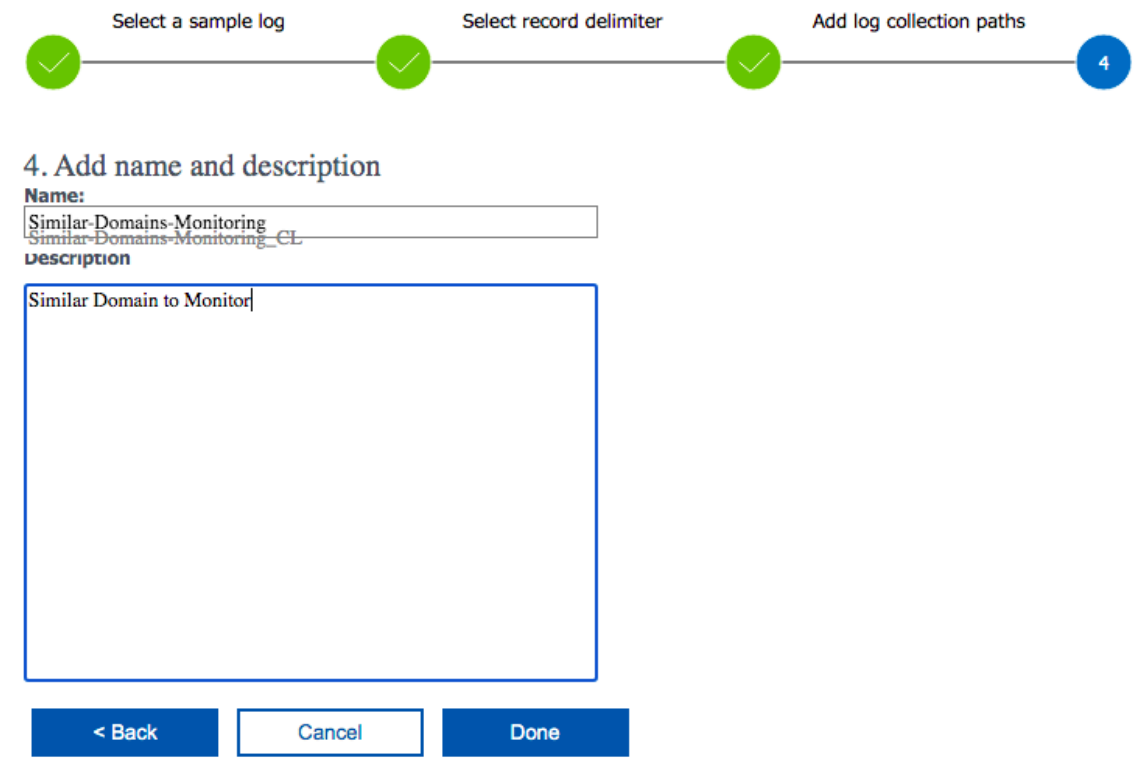
Go to "Custom logs" sections and upload a log sample (a snippet from your similar domains file)

Select the limit delimiter: New Line



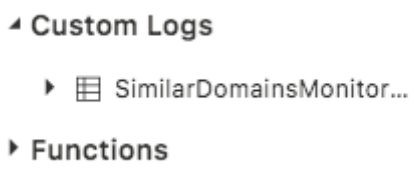
Add the file path. In my case "C:\Users\Computer\Similar-Domains.txt". If you have many log files you can use regular expression such as * eg: C:\Users\Computer*.txt

Add a name and description to your custom log source



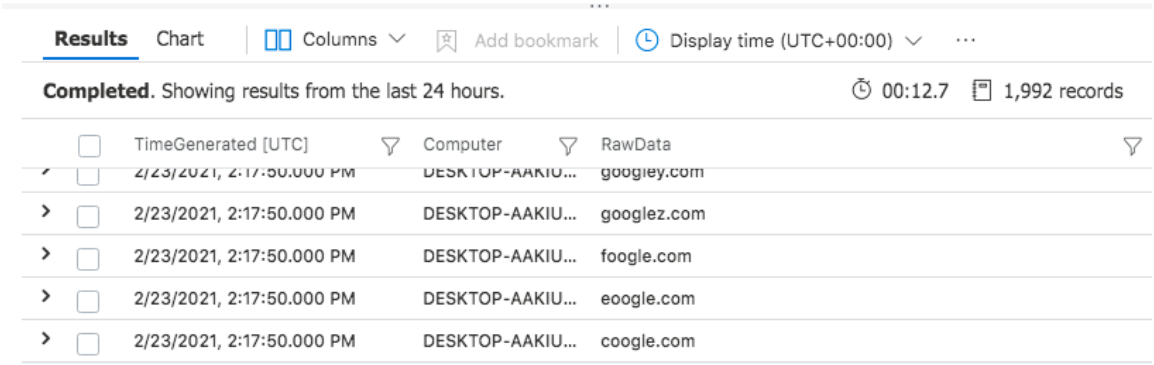
Voila! Your custom log is created successfully

Go to Sentinel log section and you will find it under **Custom Logs**



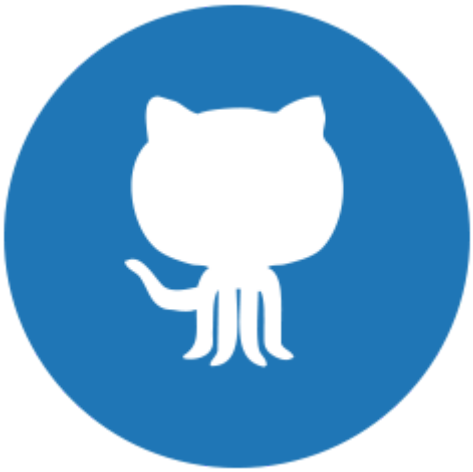
To query it, simply select its name as follows:

```
1  SimilarDomainsMonitoring_CL
```



Finally, now you can create a rule to detect if a user visited one of the similar domains. For example, you can use the JOIN function with DNSevents source.

Azure Sentinel Code Samples and Projects



- Azure Sentinel Entity Hash VirusTotal Scanner
- Azure Sentinel Report Generator
- Azure Sentinel Entity Extractor
- Azure Sentinel TheHive Playbook
- Azure Sentinel Threat Hunting Queries
- Sentinel2Attack

Azure Security Center and Security Hygiene: Small Steps, Big Impact

- Why Cyber Hygiene is important?

> “Great things are done by a series of small things brought together.“ - Vincent Van Gogh

Modern organizations face cyber threats on a daily basis. Black hat hackers do not show any indication that they are going to stop. Thus, it is essential for every organization to protect its assets and its clients against these threats. Information security is a journey and cannot be achieved overnight. Furthermore, organizations do not need the next {AI-ML-Nextgen-blockchain- put any buzzword here} security product to secure your organization, but if you need to protect your organization and users, it is essential to take the first steps. Small actions can take you so far in your cybersecurity journey.

Do you have an idea how many data breaches and cyber-attacks could be avoided by taking small actions like simply enabling MFA or by updating and patching a system?

That is why ***“Security Hygiene”*** is very important. Security hygiene is simply a set of small actions and best practices that can be performed to protect the organization and enhance its security posture. There are many security hygienes principles that you can follow immediately. Some of them are the following:

- Patching and updating systems
- Enabling MFA
- Asset Inventory and management
- User awareness and education
- Privileged accounts protection

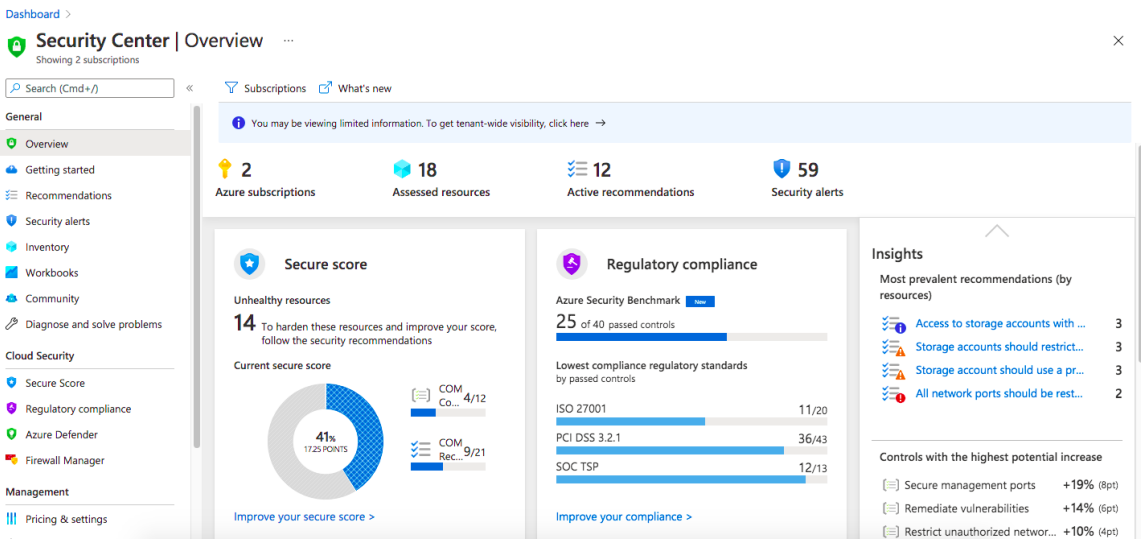
- Installing AV solutions
- Maintaining a cybersecurity policy

- Security Hygiene with Azure Security Center

> "I am always doing what I cannot do yet, in order to learn how to do it." - Vincent Van Gogh

Now let's explore how Azure Security Center can help you in your cyber hygiene journey. Microsoft documentation describes Azure Security Center as follows:

> "Azure Security Center is a unified infrastructure security management system that strengthens the security posture of your data centers, and provides advanced threat protection across your hybrid workloads in the cloud - whether they're in Azure or not - as well as on-premises."

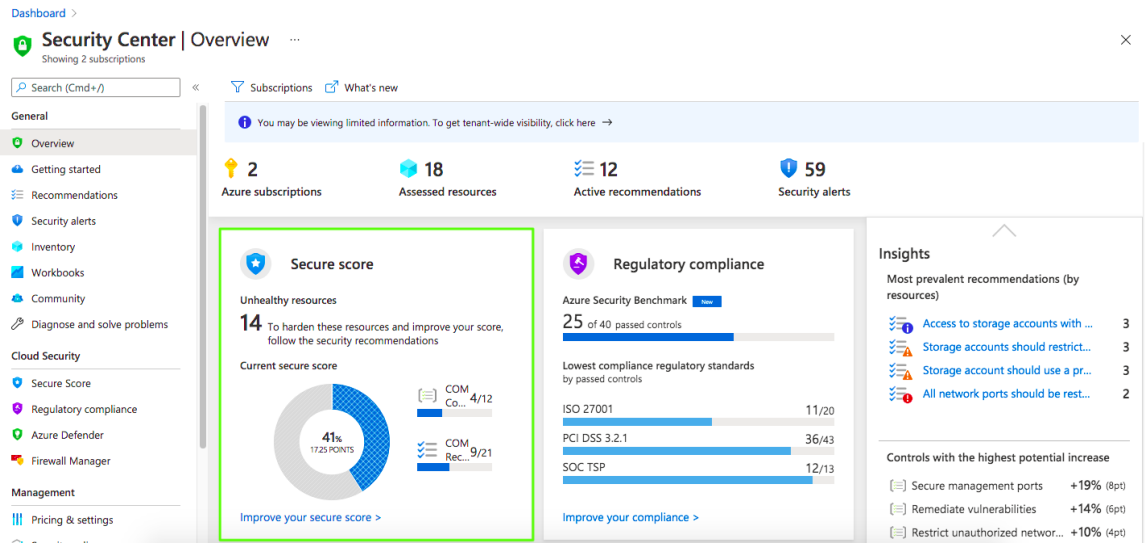


Secure Score

You can't enhance what you can't measure. That is why one of the most helpful metrics provided by the Security Center is "Secure Score". Secure Score is an aggregation of many values and assessment results to give you a clear idea about your current security situation and per consequence to help you track your situation. The score is represented as a percentage and it is calculated as follows:

$$Secure\ score\ for\ a\ subscription\ = \frac{\sum current\ scores\ for\ all\ controls}{\sum maximum\ scores\ for\ all\ controls}\ x\ 100$$

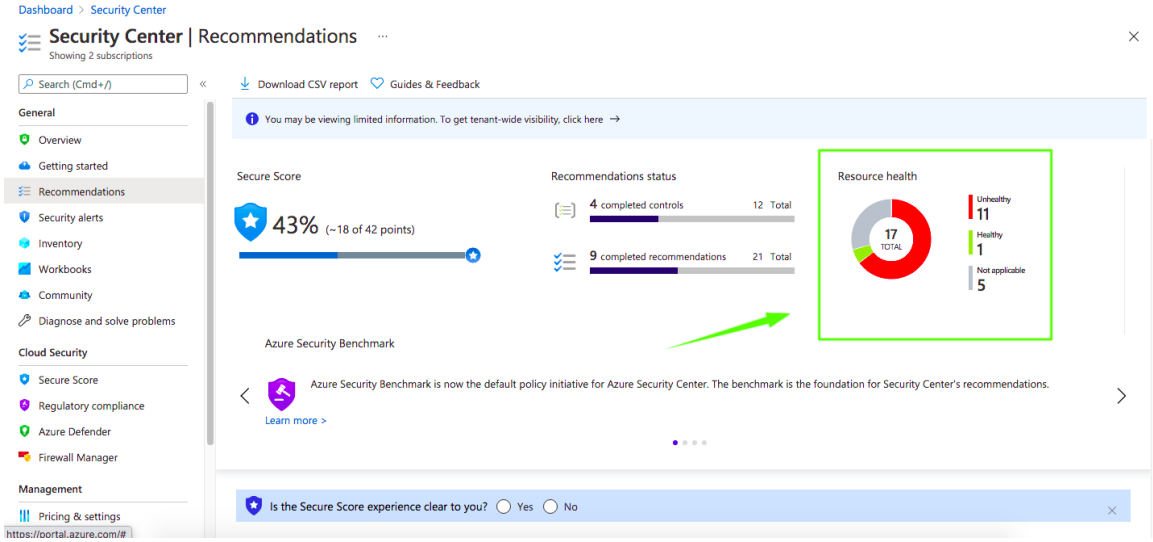
(Source)



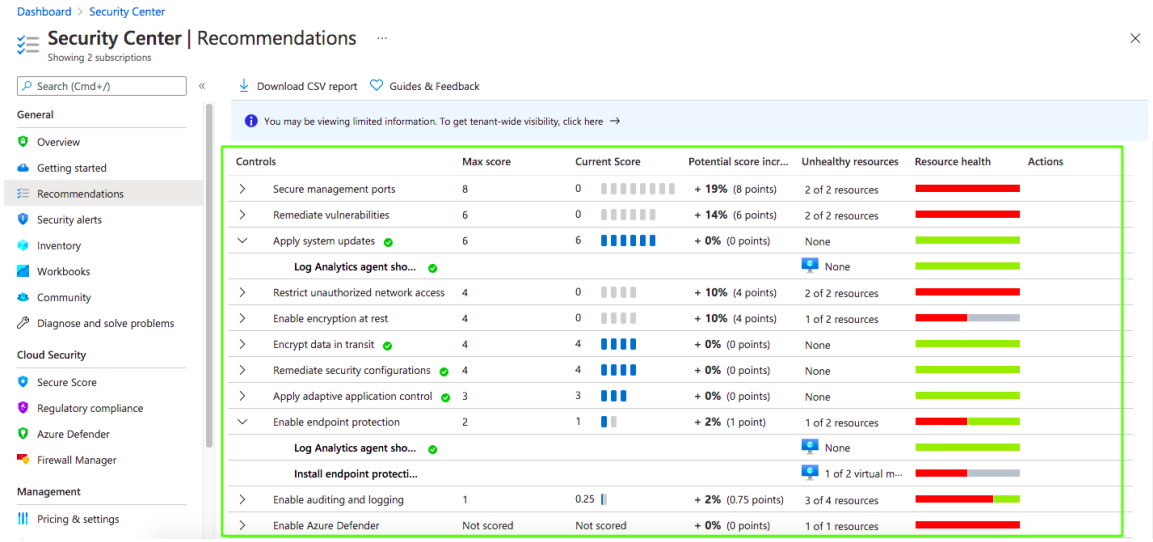
To raise the "secure score", you need to take actions based on the provided recommendations. For example, if you enable MFA, 10 points will be added to your score. More details about the scure score calculation can be found here: https://docs.microsoft.com/en-us/azure/security-center/secure-score-security-controls

Recommendations

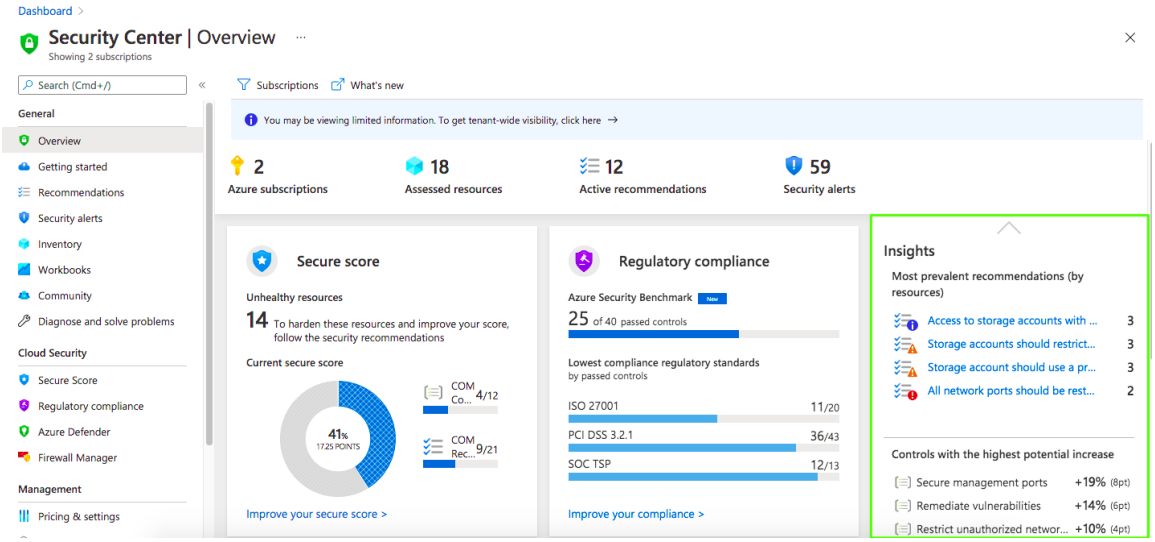
Recommendations can be found simply by selecting the "Recommendations" link in the side menu. The recommendations page gives you helpful insights about your resource health.



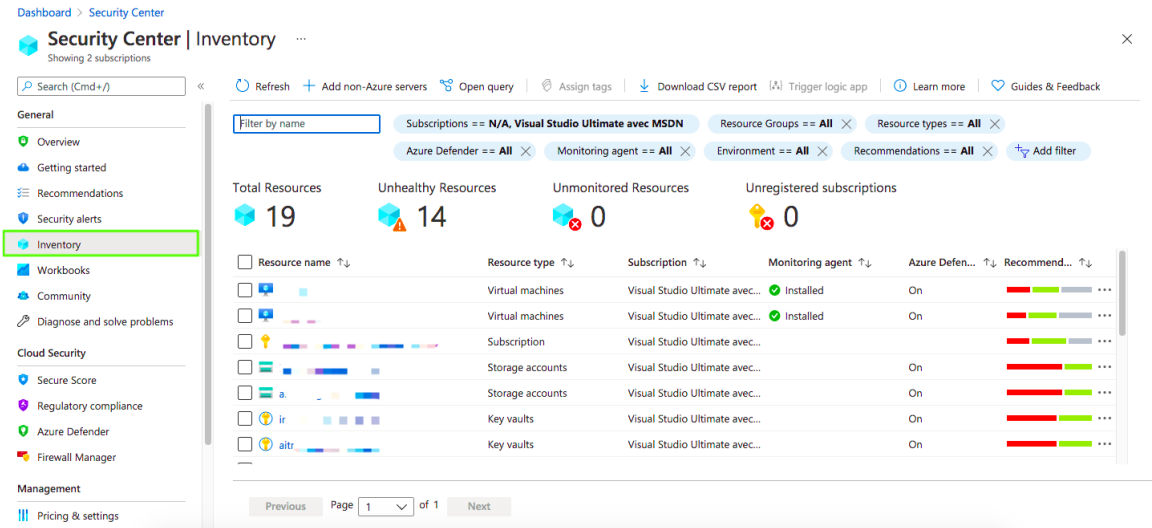
Resource health is identified based on a pre-defined list of security controls. You need to remediate the provided security controls to increase the “Secure score”. Thus your security posture will increase accordingly.



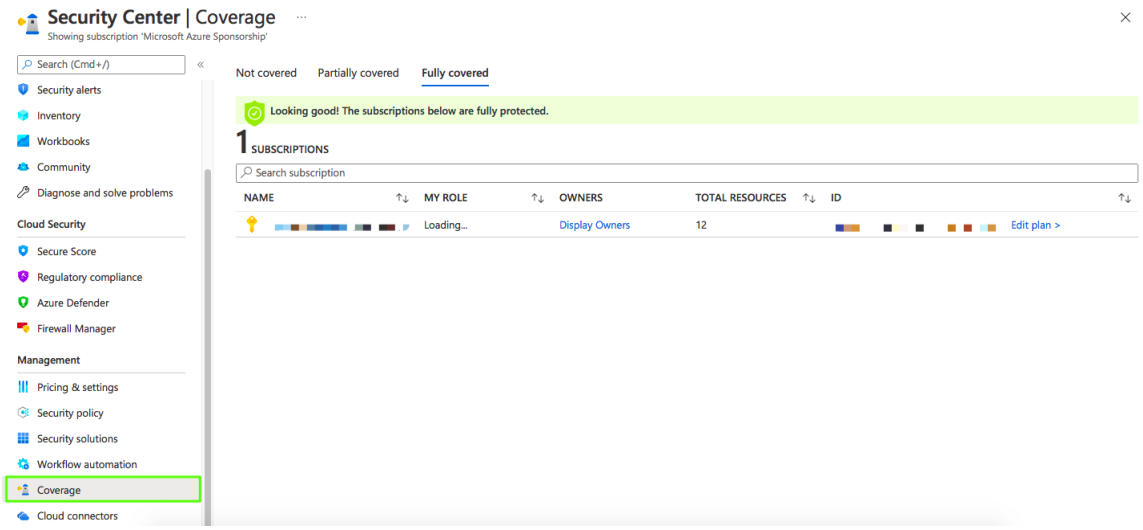
Some insights about the recommendations are shown on the main page of the security center



Visibility is very important when it comes to information security and especially in security hygiene. Azure Security Center gives you clear visibility for your assets and resources on the “Inventory” page.

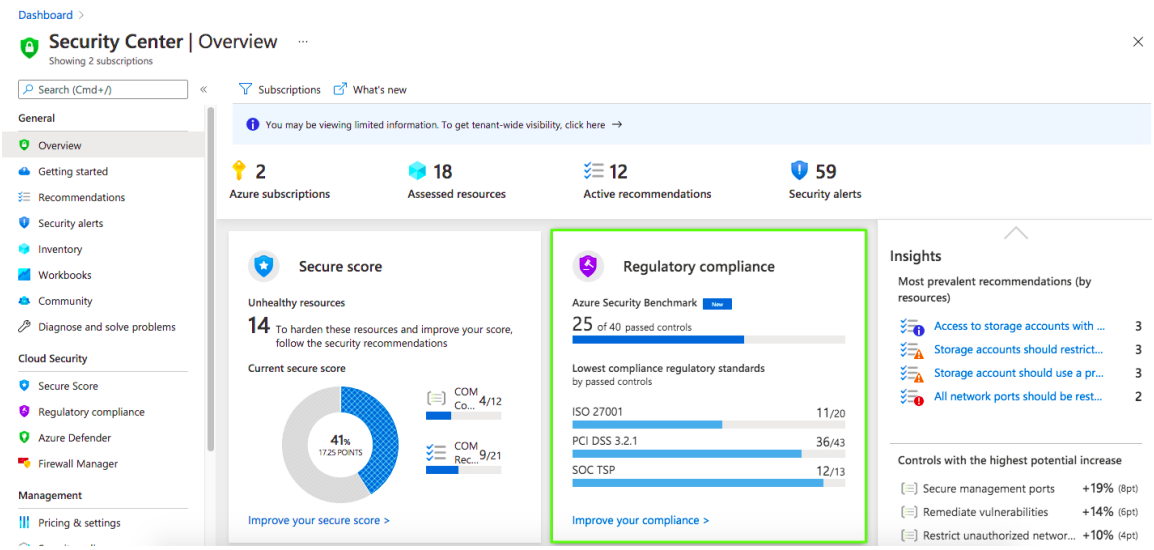


Furthermore, it is possible to check the coverage by exploring the “coverage” page, where you can identify the covered Azure subscriptions.



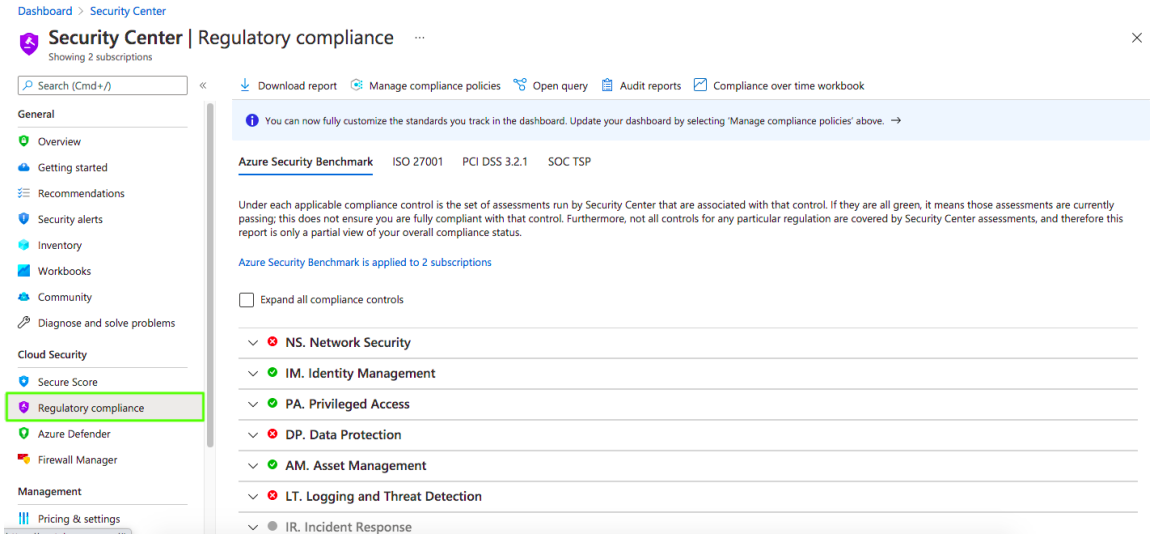
Regulatory Compliance

Many organizations need to be aligned and compliant with industry and regulatory standards, and benchmarks. Azure Security Center saves your precious time and provides you with a regulatory compliance section where you can ensure how your organization is aligned with industry standards or internal policies.

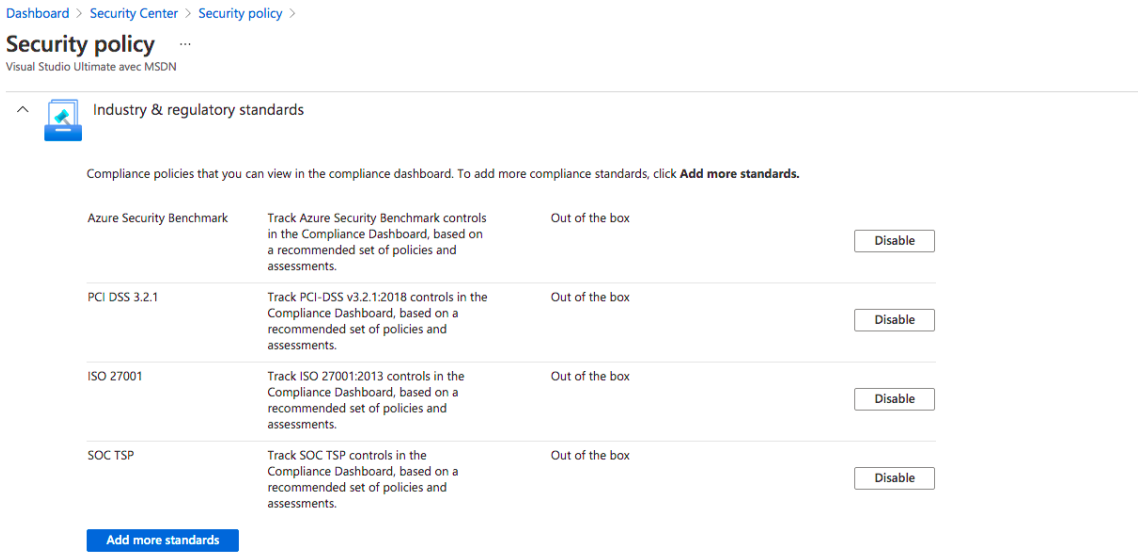


To explore it, simply select the “Regulatory compliance” page. For example, as a start, you are provided with “Azure Security Benchmark v2”.

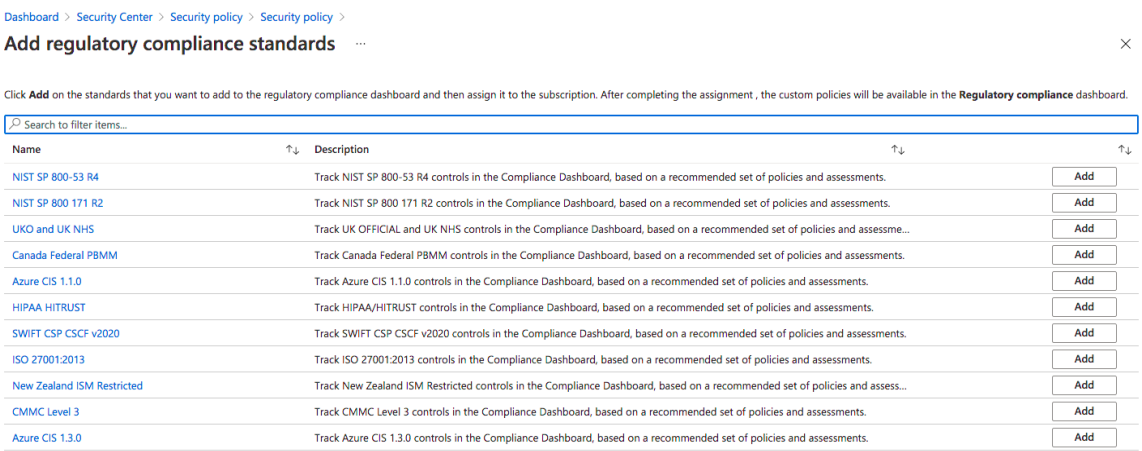
> “The Azure Security Benchmark (ASB) provides prescriptive best practices and recommendations to help improve the security of workloads, data, and services on Azure.” (Source: https://docs.microsoft.com/en-us/security/benchmark/azure/overview)



You can enable and disable the standards



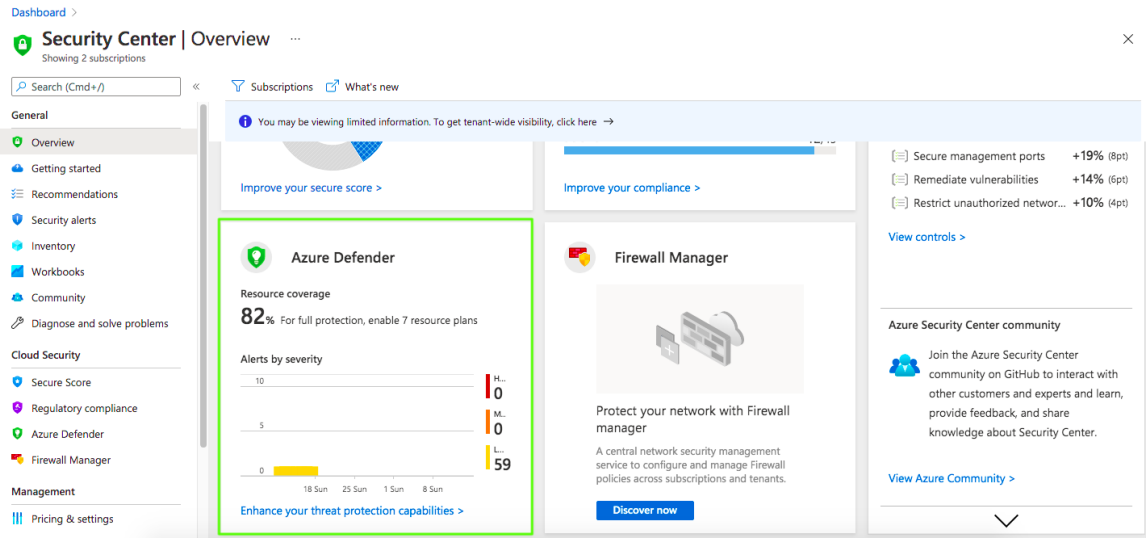
Furthermore, you can add regulatory compliance standards from a list provided by the security center to help you start right away.



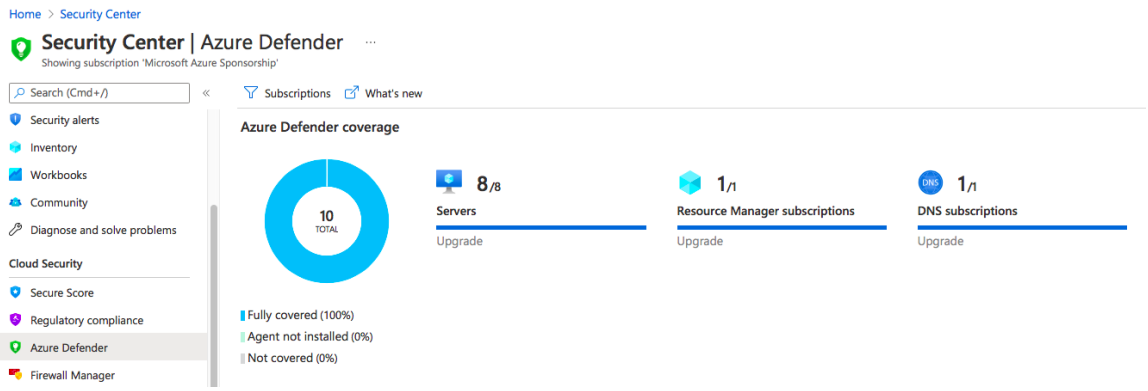
Azure Defender

Azure defender is integrated with the Security center and it helps you protect your hybrid resources and workloads. According to Microsoft documentation:

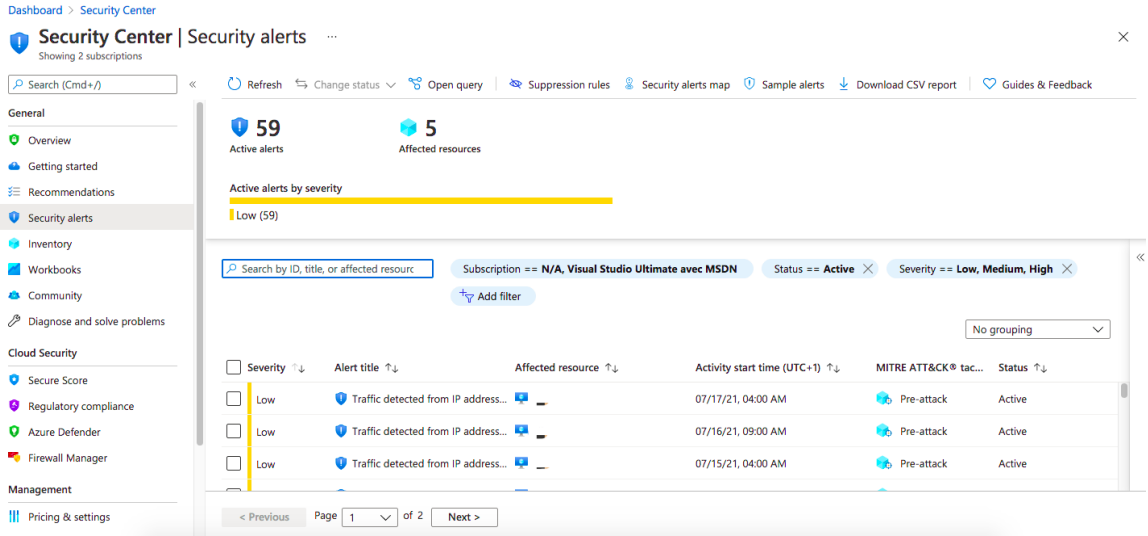
“Azure Defender provides security alerts and advanced threat protection for virtual machines, SQL databases, containers, web applications, your network, and more.”



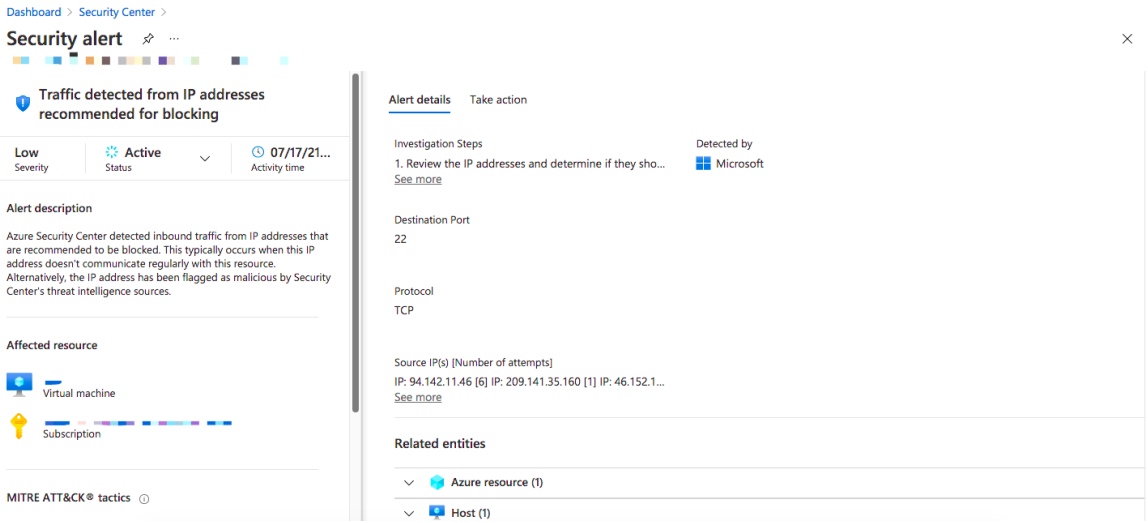
Azure Defender is not enabled per default.



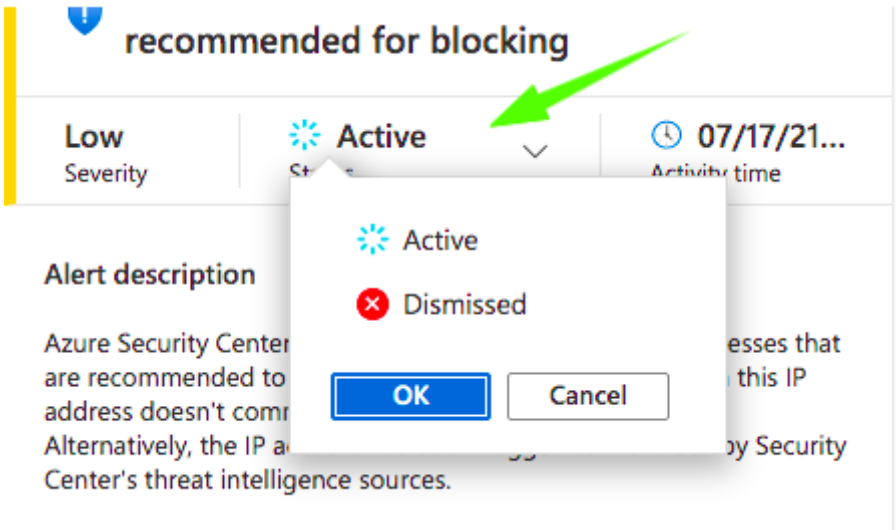
Alerts are shown on the “Security Alerts” page where you can see the triggered alerts with different severities and the affected resources.



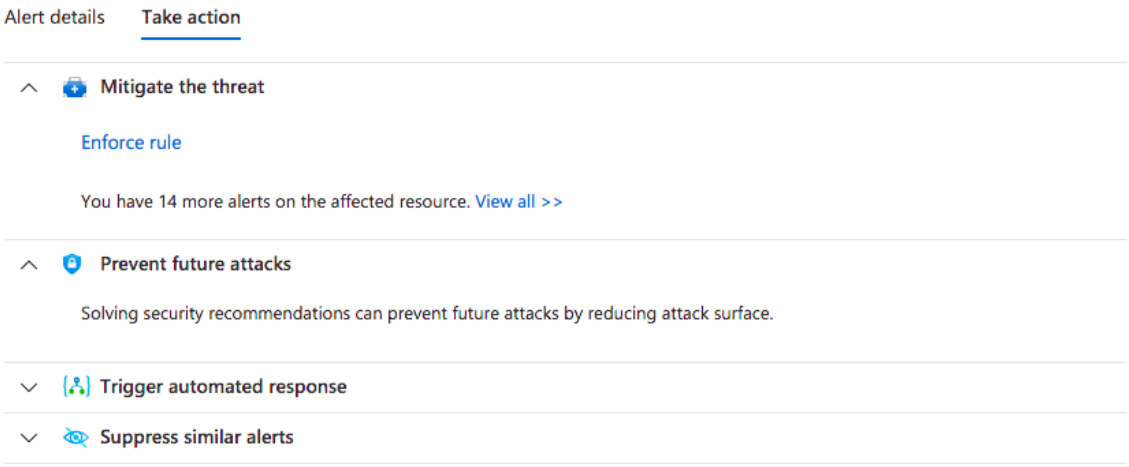
If you select a specific alert you will get more details about it



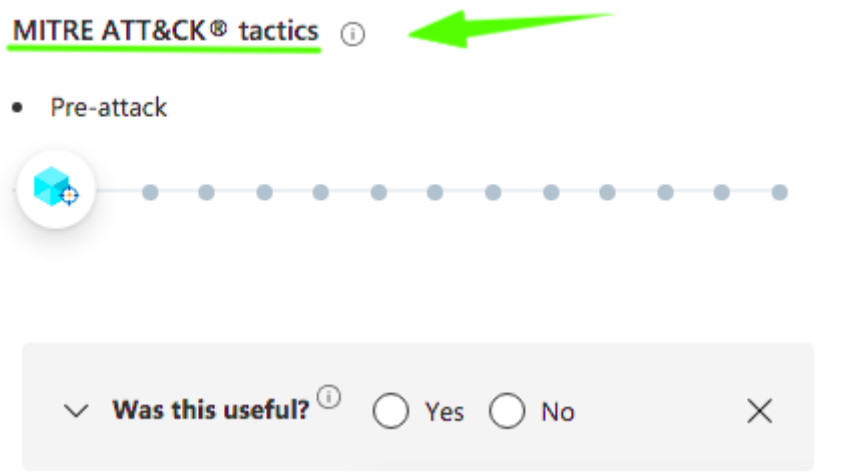
Alert status can be changed by clicking on the status option:



Not only alert details are presented. The “take action” option gives you the ability to mitigate the threats and even trigger automated tasks.



Alerts are mapped to the MITRE ATT&CK Framework. MITRE ATT&CK is a framework developed by the Mitre Corporation. The comprehensive document classifies adversary attacks, in other words, their techniques and tactics after observing millions of real-world attacks against many different organizations. This is why ATT&CK refers to "Adversarial Tactics, Techniques & Common Knowledge".



Nowadays the frameworks provide different matrices: Enterprise, Mobile, and PRE-ATT&CK. Each matrix contains different tactics and each tactic has many techniques.

ATT&CK®

Tactics, Techniques, and procedures (TTPs) are how the attackers are going to achieve their mission. A tactic is the highest level of attack behaviour. The PRE-ATT&CK MITRE framework

present the 15 tactics as the following:

1. Priority Definition Planning
2. Priority Definition Direction
3. Target Selection
4. Technical Information Gathering
5. People Information Gathering
6. Organizational Information Gathering
7. Technical Weakness Identification
8. People Weakness Identification
9. Organizational Weakness Identification
10. Adversary OPSEC
11. Establish & Maintain Infrastructure
12. Persona Development
13. Build Capabilities
14. Test Capabilities
15. Stage Capabilities

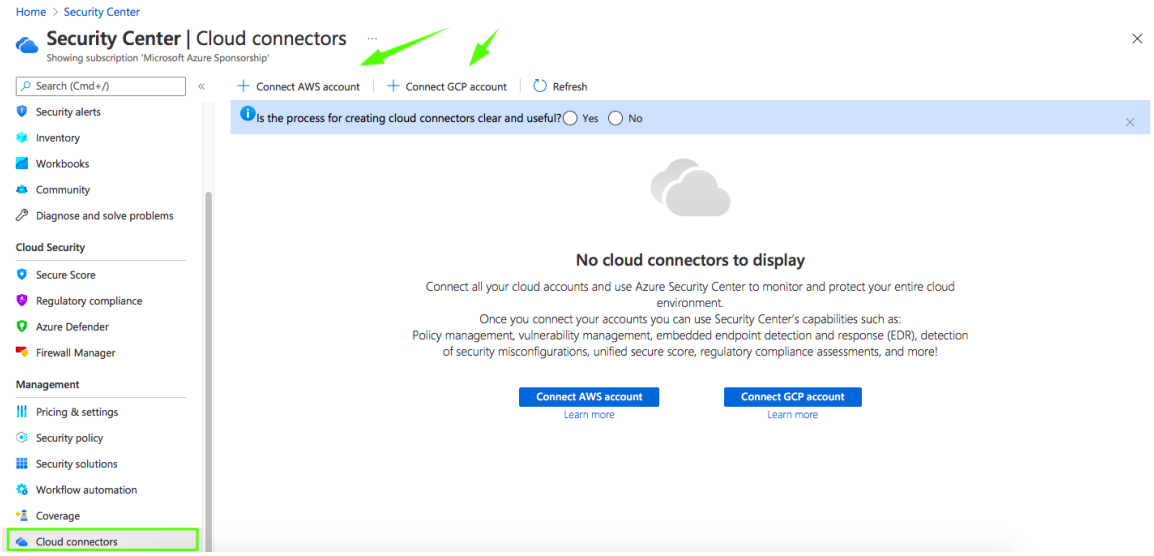
PRE-ATT&CK Matrix

Below are the tactics and techniques representing the MITRE PRE-ATT&CK Matrix.

Last Modified: 2018-04-18 17:59:24.739000
version permalink

Priority Definition Planning	Priority Definition Direction	Target Selection	Technical Information Gathering	People Information Gathering	Organizational Information Gathering	Technical Weakness Identification	People Weakness Identification	Organizational Weakness Identification	Adversary OPSEC	Establish & Maintain Infrastructure	Persona Development	Build Capabilities	Test Capabilities
Assess current holdings, needs, and wants	Assign KITs, KIQs, and/or intelligence requirements	Determine approach/attack vector	Acquire OSINT data sets and information	Acquire OSINT data sets and information	Acquire OSINT data sets and information	Analyze application security posture	Analyze organizational skillsets and deficiencies	Analyze business processes	Acquire and/or use 3rd party infrastructure services	Acquire and/or use 3rd party infrastructure services	Build social network persona	Build and configure delivery systems	Review logs and residual traces
Assess KITs/KIQs benefits	Receive KITs/KIQs and determine requirements	Determine highest level tactical element	Conduct active scanning	Aggregate individual's digital footprint	Conduct social engineering	Analyze architecture and configuration posture	Analyze social and business relationships, interests, and affiliations	Analyze organizational skillsets and deficiencies	Acquire and/or use 3rd party software services	Acquire and/or use 3rd party software services	Choose pre-compromised mobile app developer account credentials or signing keys	Build or acquire exploits	Test ability to evade automated mobile application security analysis performed by app stores
Assess leadership areas of interest	Submit KITs, KIQs, and intelligence requirements	Determine operational element	Conduct passive scanning	Conduct social engineering	Determine 3rd party infrastructure services	Analyze data collected	Assess targeting options	Analyze presence of outsourced capabilities	Acquire or compromise 3rd party signing certificates	Acquire or compromise 3rd party signing certificates	Choose pre-compromised persona and affiliated accounts	C2 protocol development	Test callback functionality
Assign KITs/KIQs into categories	Task requirements	Determine secondary level tactical element	Conduct social engineering	Identify business relationships	Determine centralization of IT management	Analyze hardware/software security defensive capabilities		Assess opportunities created by business deals	Anonymity services	Buy domain name	Develop social network persona digital footprint	Compromise 3rd party or closed-source vulnerability/exploit information	Test malware in various execution environments
Conduct cost/benefit analysis		Determine strategic target	Determine 3rd party infrastructure services	Identify groups/roles	Determine physical locations	Analyze organizational skillsets and deficiencies		Assess security posture of physical locations	Common, high volume protocols and software	Compromise 3rd party infrastructure to support delivery	Friend/Follow/Connect to targets of interest	Create custom payloads	Test malware to evade detection
Create implementation plan			Determine domain and IP address space	Identify job postings and needs/gaps	Dumpster dive	Identify vulnerabilities in third-party software libraries		Assess vulnerability of 3rd party vendors	Compromise 3rd party infrastructure to support delivery	Create backup infrastructure	Obtain Apple iOS enterprise distribution key pair and certificate	Create infected removable media	Test physical access

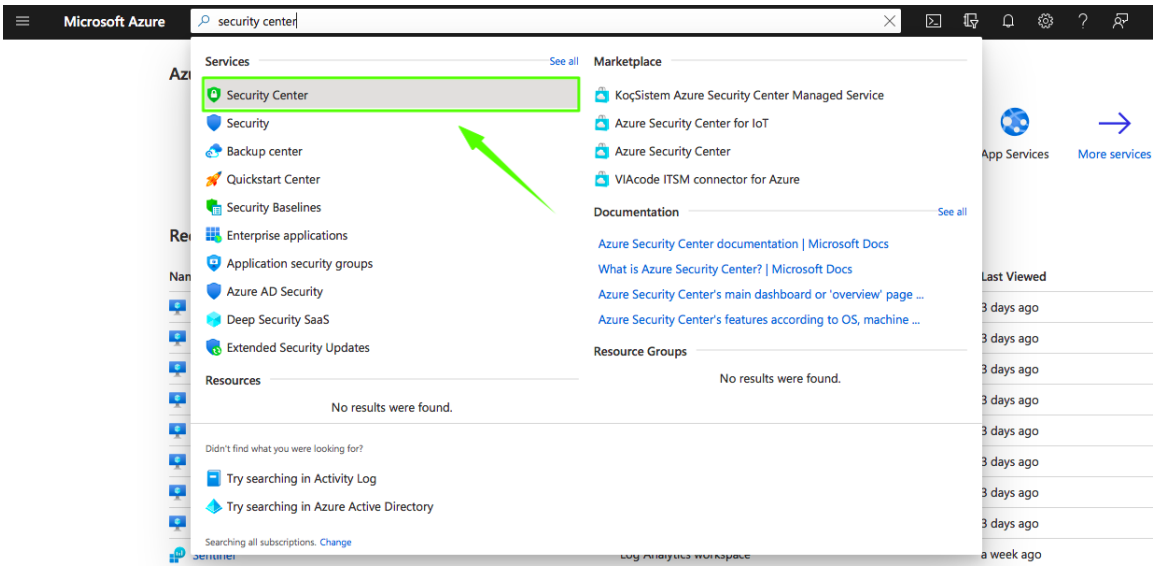
Azure Security Center gives you the ability to integrate workloads from other cloud providers such as AWS and Google GCP. To connect your cloud accounts select the “Cloud Connectors” page.



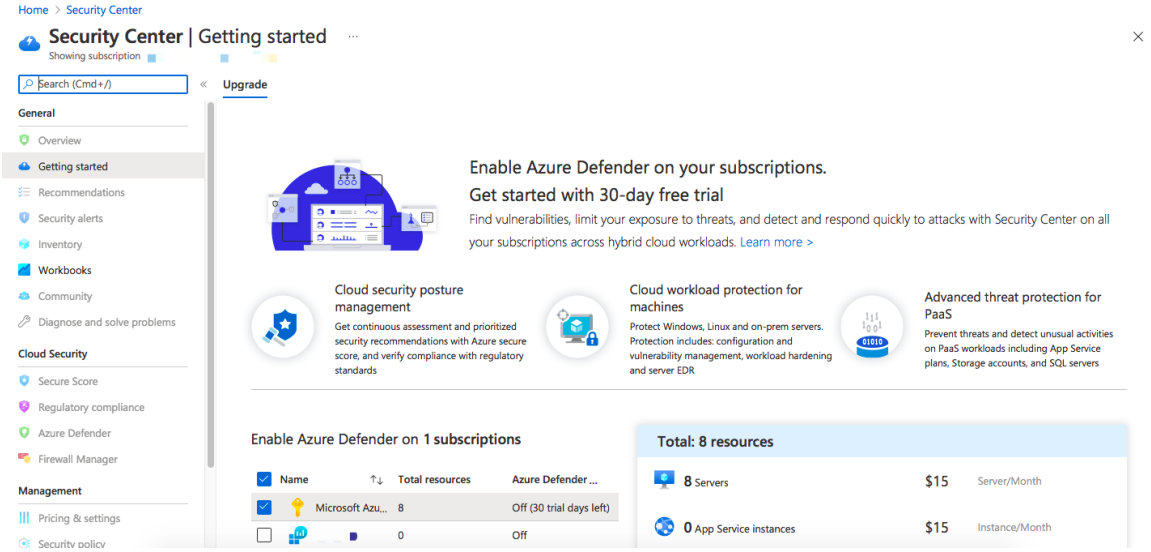
- Take Actions Now

> "What would life be if we had no courage to attempt anything?" - Vincent Van Gogh

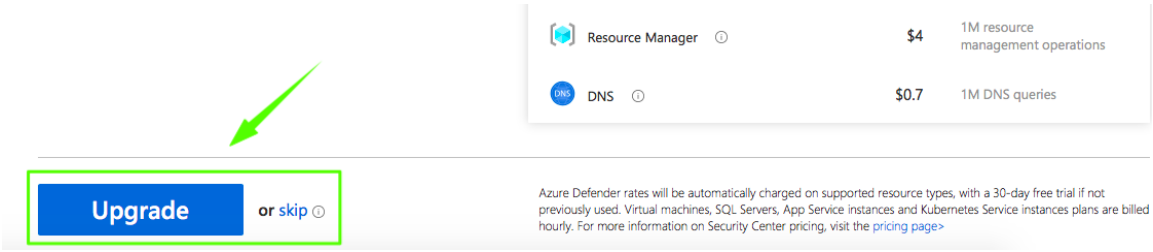
It is time to take some actions and try Azure Security Center by yourself. Go to your Azure Portal and search for "Security Center"



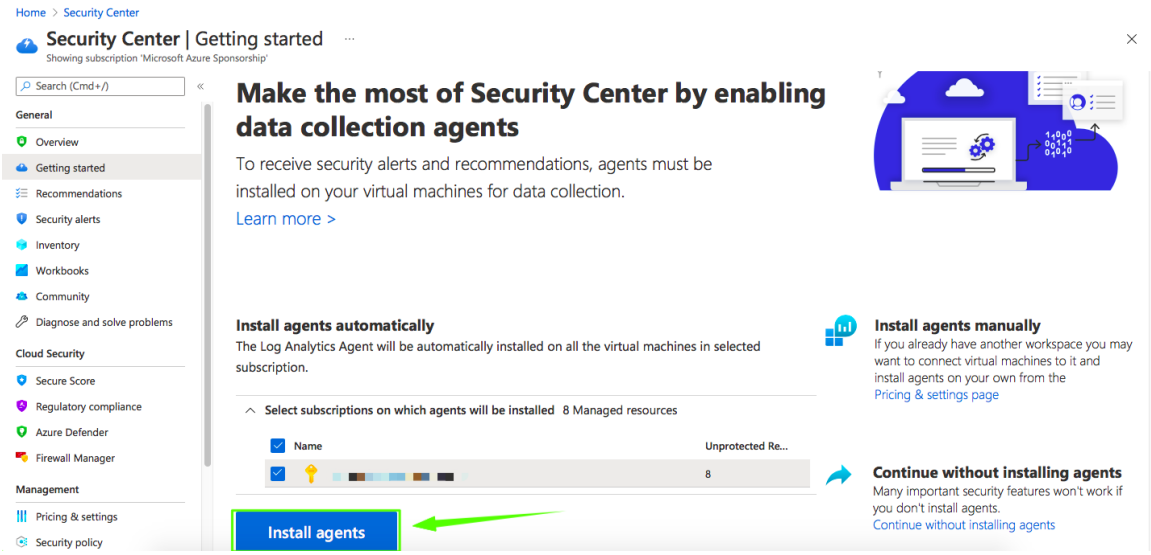
You will be taken to the "Getting Started Page"



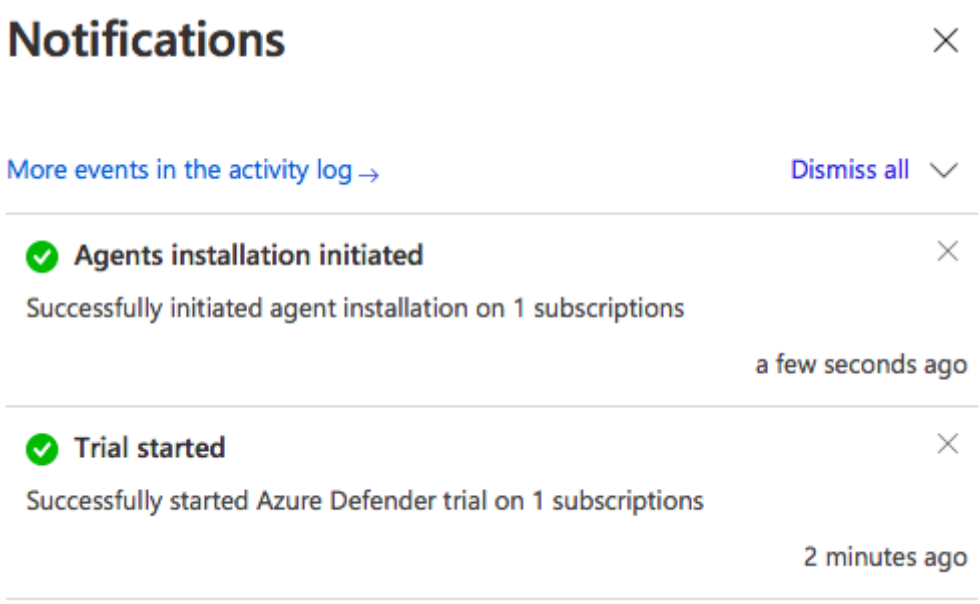
Click on upgrade to start a 30-day free trial



Click on "Install Agents"



Voila! Now you can start exploring Azure Security Center



Now go to the recommendations page and try to raise the "Secure Score"