



HideZeroOne
INE – Cyber Sec
www.hide01.ir





Incident Handling & Response Professional

Intrusion Detection By Analyzing Traffic (Part 2)

Section 02 | Module 02

v1

© Caendra Inc. 2018
All Rights Reserved

OUTLINE

Section 2 | Module 2: Intrusion
Detection by Analyzing Traffic - Part 2

Table of Contents

Learning Objectives

▼ 2.1 Analyzing & Detecting Transport
Layer Attacks

2.1 Analyzing & Detecting
Transport Layer Attacks

▼ 2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

Table of Contents

Module 02 – Intrusion Detection By Analyzing Traffic (Part 2)

2.1 Analyzing & Detecting Transport Layer Attacks

2.2 Analyzing Common Application Protocol Traffic & Attacks

2.3 Effectively Using Open Source IDS (Lab-based)

OUTLINE

Section 2 | Module 2: Intrusion
Detection by Analyzing Traffic - Part 2

Table of Contents

Learning Objectives

▼ 2.1 Analyzing & Detecting Transport Layer Attacks

2.1 Analyzing & Detecting
Transport Layer Attacks

▼ 2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

Learning Objectives

By the end of this module, you should have a better understanding of:

- ✓ How the attacks at the Transport and Application layers work
- ✓ How to detect attacks at the Transport and Application layers
- ✓ The detection capabilities of open-source IDS

OUTLINE

Section 2 | Module 2: Intrusion Detection by Analyzing Traffic - Part 2

Table of Contents

Learning Objectives

▼ 2.1 Analyzing & Detecting Transport Layer Attacks

2.1 Analyzing & Detecting Transport Layer Attacks

▼ 2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

IHRP

2.1

Analyzing & Detecting Transport Layer Attacks



OUTLINE

Section 2 | Module 2: Intrusion
Detection by Analyzing Traffic - Part 2

Table of Contents

Learning Objectives

▼ 2.1 Analyzing & Detecting Transport
Layer Attacks

2.1 Analyzing & Detecting
Transport Layer Attacks

▼ 2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1 Analyzing & Detecting Transport Layer Attacks

Up to this point, we have seen how to analyze and detect attacks at the IEEE 802.x Link layer.

Let's now cover how to analyze and detect attacks at the Transport layer. We'll focus on the TCP, UDP and ICMP protocols.

OUTLINE

Section 2 | Module 2: Intrusion Detection by Analyzing Traffic - Part 2

Table of Contents

Learning Objectives

▼ 2.1 Analyzing & Detecting Transport Layer Attacks

2.1 Analyzing & Detecting Transport Layer Attacks

▼ 2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

Let's start our journey to understanding the Transport layer, by analyzing the most important aspects of the TCP protocol.

TCP is a reliable and connection-oriented protocol, leveraged by many application-layer protocols to deliver packets.

OUTLINE

Section 2 | Module 2: Intrusion Detection by Analyzing Traffic - Part 2

Table of Contents

Learning Objectives

▼ 2.1 Analyzing & Detecting Transport Layer Attacks

2.1 Analyzing & Detecting Transport Layer Attacks

▼ 2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

TCP is quite complex, due to its rich functionality. This is apparent when studying the TCP header.

OUTLINE

Section 2 | Module 2: Intrusion
Detection by Analyzing Traffic - Part 2

Table of Contents

Learning Objectives

▼ 2.1 Analyzing & Detecting Transport
Layer Attacks

2.1 Analyzing & Detecting
Transport Layer Attacks

▼ 2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

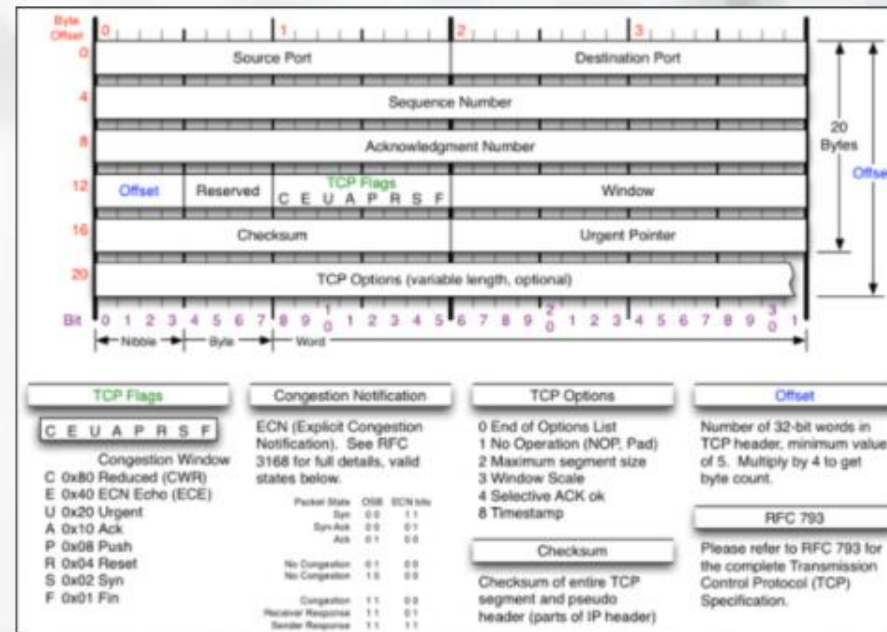
2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

The TCP header contains information such as:

- **Source Port:** The port from where the packet is transmitted
- **Destination Port:** The port where the packet will arrive
- **Sequence Number:** Field value that ensures the integrity of a data stream. It identifies a TCP segment.
- **Acknowledgement Number:** Field value containing the expected sequence number of the next packet to arrive from the connection peer.
- **Flags:** Note that, the URG, ACK, PSH, RST, SYN and FIN flags are used to identify the type of the packet in transit
- **Window Size:** Indicates the TCP receiver buffer size
- **Checksum:** Field value that ensures the integrity of the TCP header and data upon arrival
- **Urgent Pointer:** In case the URG flag is set, this field indicates the position within the packet from where reading data should start
- **Options:** Optional fields that may be specified



Source: <https://skminhaj.wordpress.com/2016/02/15/tcp-segment-vs-udp-datagram-header-format/>

OUTLINE

Section 2 | Module 2: Intrusion Detection by Analyzing Traffic - Part 2

Table of Contents

Learning Objectives

▼ 2.1 Analyzing & Detecting Transport Layer Attacks

2.1 Analyzing & Detecting Transport Layer Attacks

▼ 2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

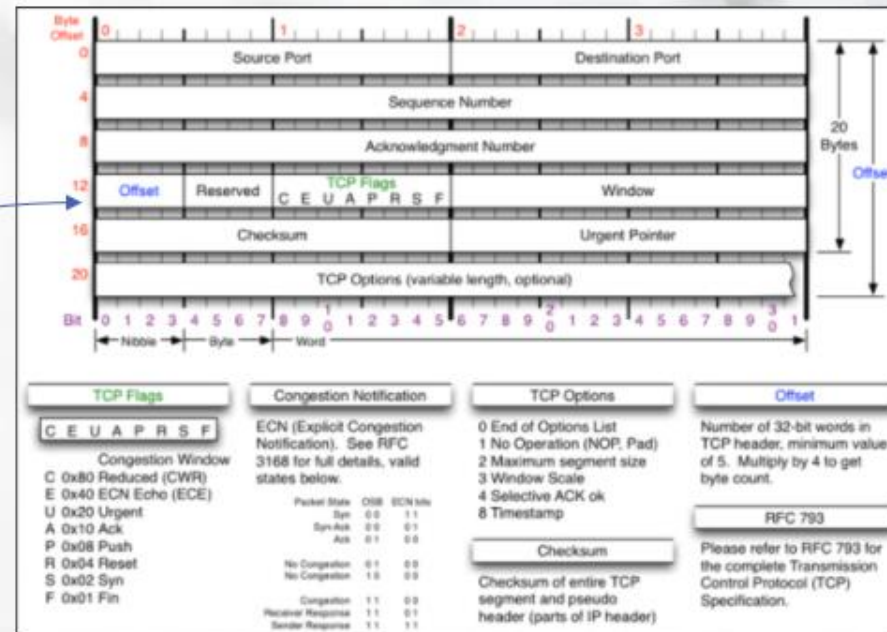
2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

Notes:

1. The TCP header's size is 20 bytes, when no options are specified
2. Offset = TCP header length
3. When the SYN flag is set on a TCP segment, you'll most probably come across the TCP maximum segment size option
4. When the SYN flag is set, there are no data following the header



Source: <https://skminhaj.wordpress.com/2016/02/15/tcp-segment-vs-udp-datagram-header-format/>

OUTLINE

Section 2 | Module 2: Intrusion Detection by Analyzing Traffic - Part 2

Table of Contents

Learning Objectives

▼ 2.1 Analyzing & Detecting Transport Layer Attacks

2.1 Analyzing & Detecting Transport Layer Attacks

▼ 2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

As already mentioned, TCP is a connection-oriented protocol. For a connection to be established between two hosts, a procedure known as *3-way handshake* is required. Then, the actual data transmission can commence.

OUTLINE

Section 2 | Module 2: Intrusion Detection by Analyzing Traffic - Part 2

Table of Contents

Learning Objectives

▼ 2.1 Analyzing & Detecting Transport Layer Attacks

2.1 Analyzing & Detecting Transport Layer Attacks

▼ 2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

The header fields involved in the handshake are:

- Sequence number
- Acknowledgement numbers
- SYN and ACK flags

```
0                               1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Source Port                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Sequence Number                          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Acknowledgment Number                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Data |                               |U|A|P|R|S|F|
| Offset| Reserved |R|C|S|S|Y|I|
|       |           |G|K|H|T|N|N|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

OUTLINE

Section 2 | Module 2: Intrusion
Detection by Analyzing Traffic - Part 2

Table of Contents

Learning Objectives

▼ 2.1 Analyzing & Detecting Transport
Layer Attacks

2.1 Analyzing & Detecting
Transport Layer Attacks

▼ 2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

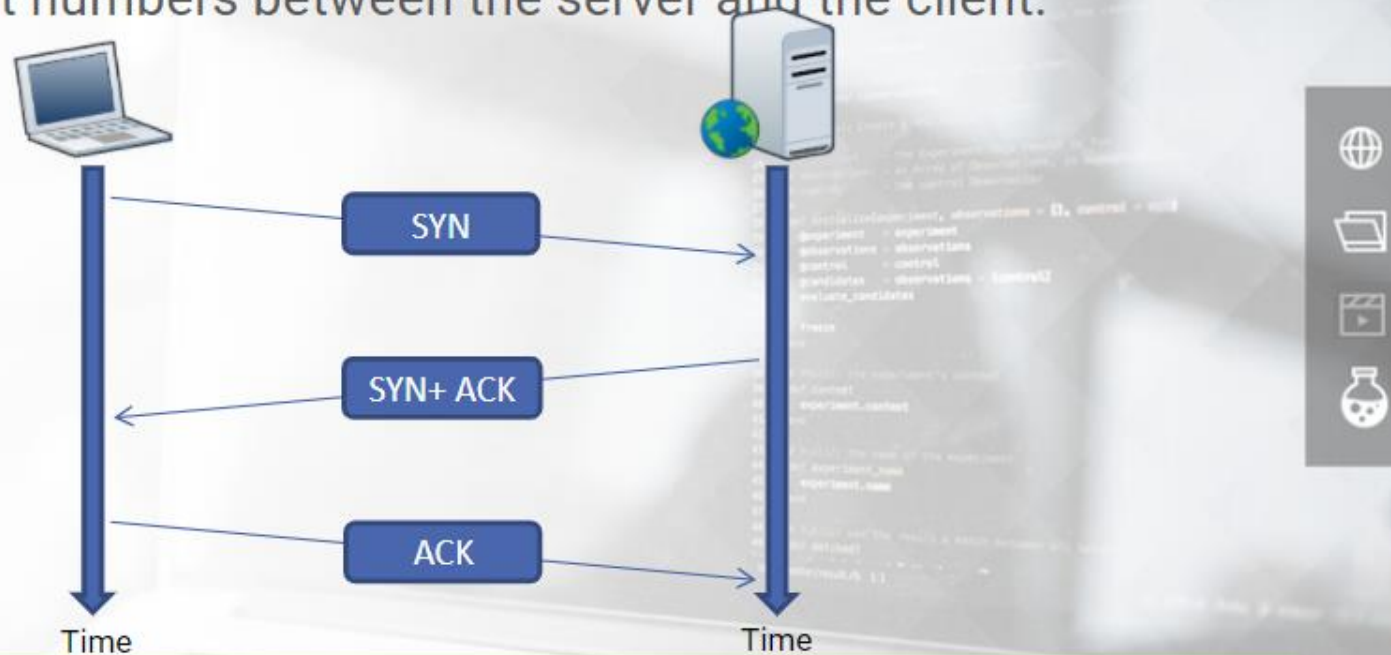
2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

The steps in the handshake are used to synchronize the sequence and acknowledgement numbers between the server and the client.



OUTLINE

Detection of Analyzing Traffic

Table of Contents

Learning Objectives

▼ 2.1 Analyzing & Detecting Transport Layer Attacks

2.1 Analyzing & Detecting Transport Layer Attacks

▼ 2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

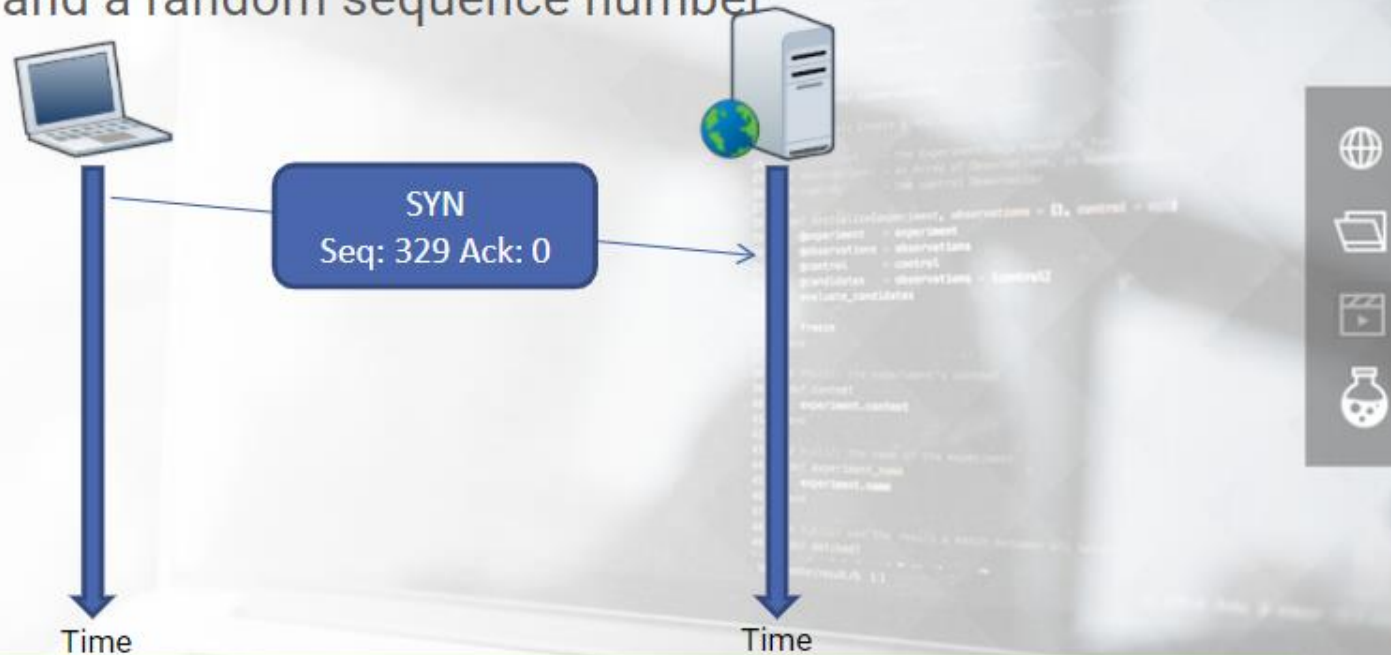
2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

During the first step, the client sends a TCP packet to the server with the SYN flag enabled and a random sequence number



OUTLINE

Learning Objectives

▼ 2.1 Analyzing & Detecting Transport Layer Attacks

2.1 Analyzing & Detecting Transport Layer Attacks

▼ 2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

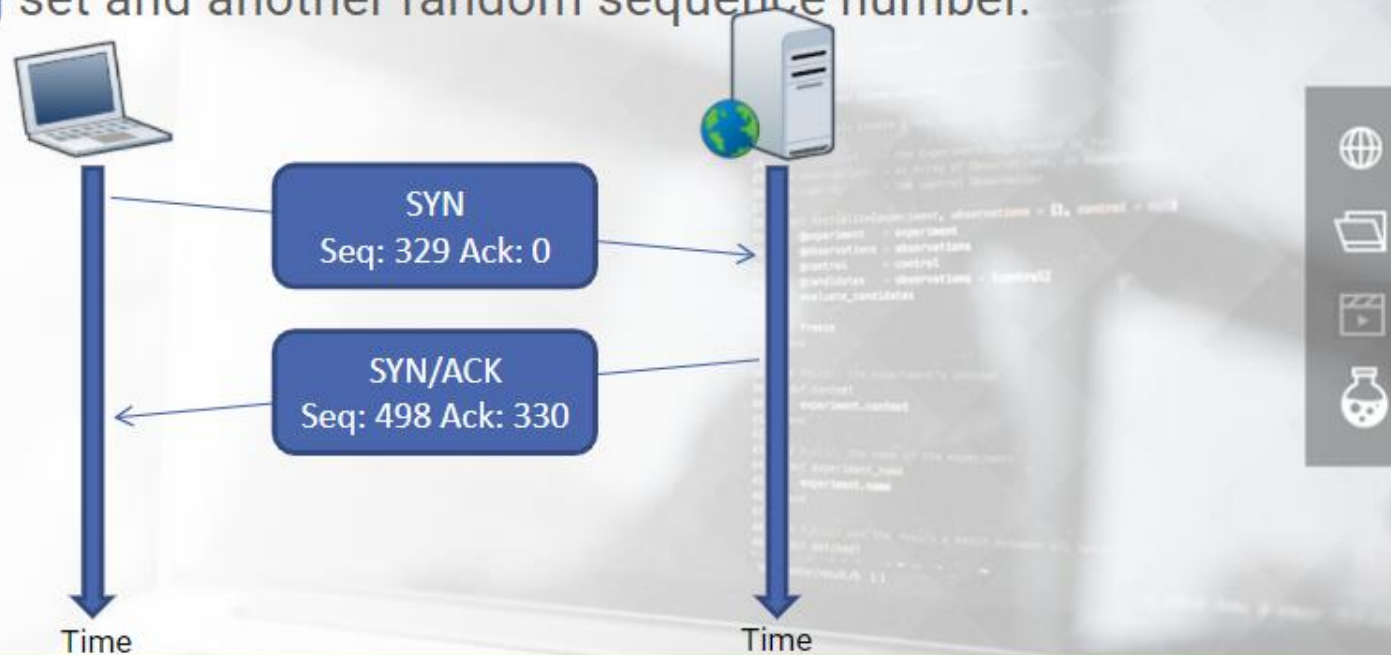
2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

In the second step, the server replies by sending a packet with both the SYN and ACK flag set and another random sequence number.



OUTLINE

▼ 2.1 Analyzing & Detecting Transport Layer Attacks

2.1 Analyzing & Detecting Transport Layer Attacks

▼ 2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

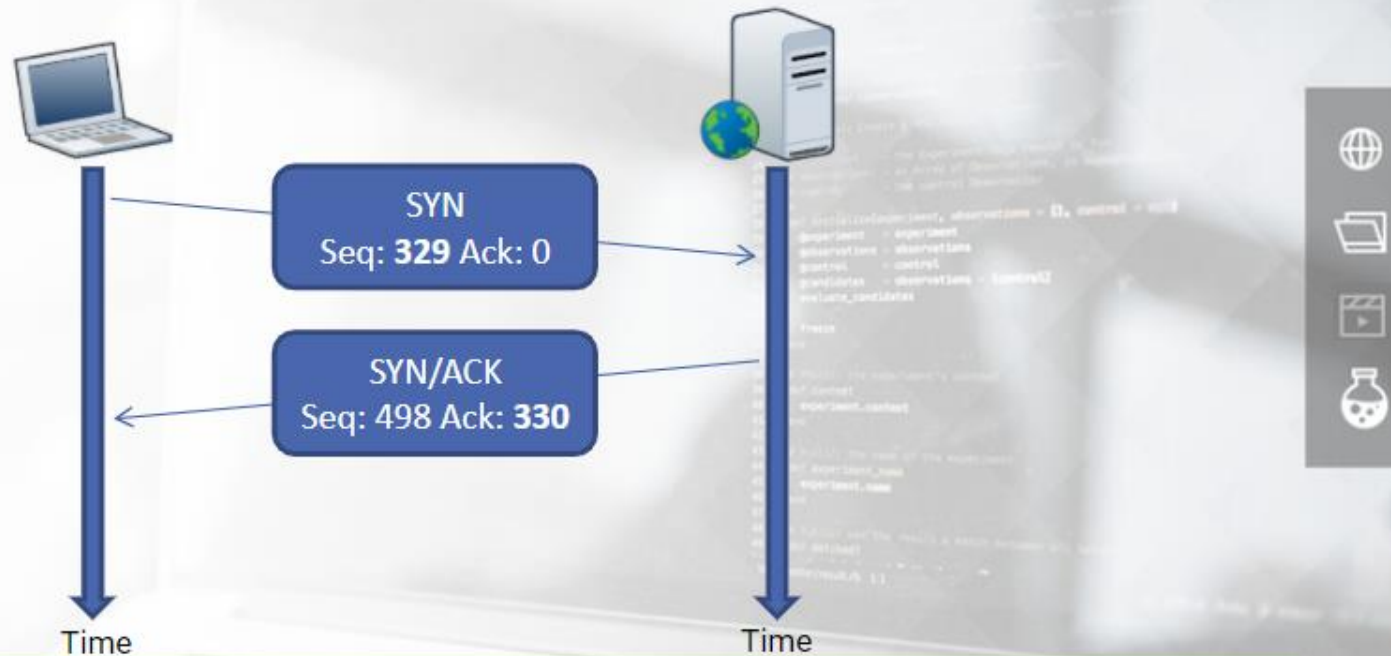
2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

The ACK number is always a simple increment of the SYN number sent by the client.



OUTLINE

Logix Firewall

2.1 Analyzing & Detecting
Transport Layer Attacks

▼ 2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

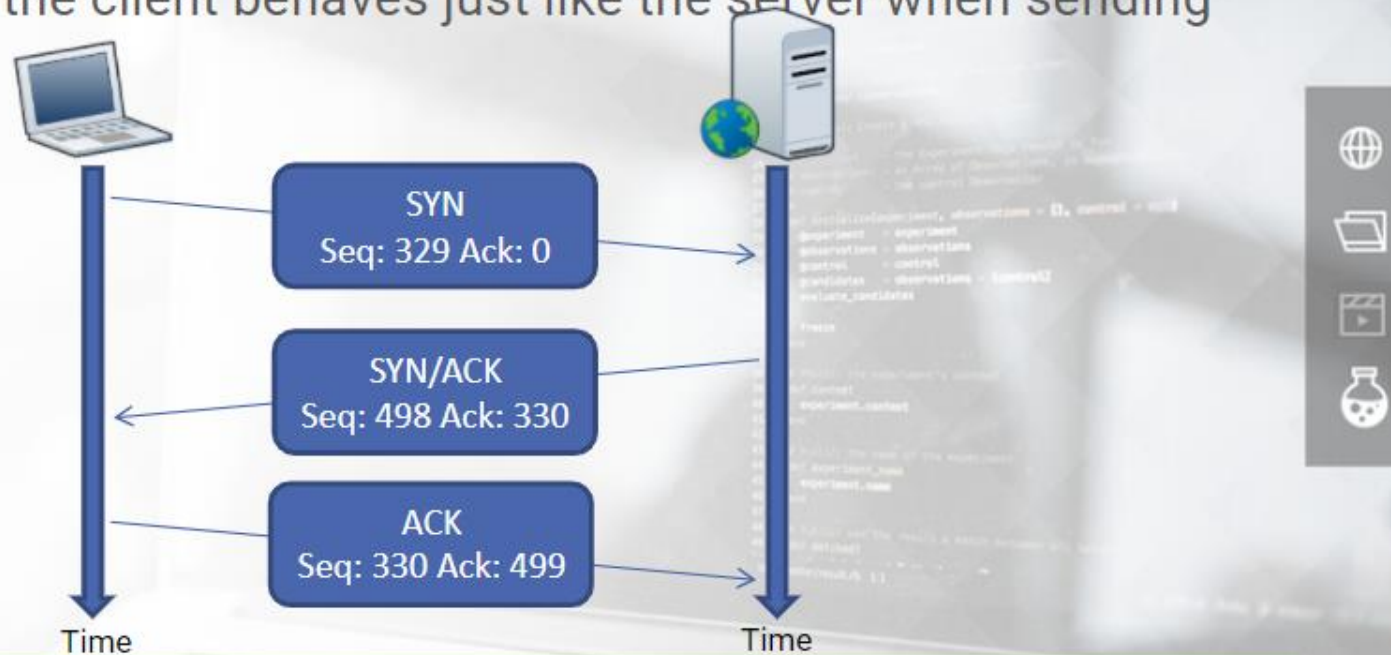
2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

Finally the client completes the synchronization by sending an ACK packet. Note that the client behaves just like the server when sending ACK packets.



OUTLINE

Transport Layer Protocols

▼ 2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

Consider the *3-way_handshake.pcap* file found in this module's resources. Let's now see, how the 3-way handshake looks like on the wire (Wireshark).

1	0.000000	192.168.1.2	174.143.213.184	TCP	74	54841 → 80 [SYN] Seq=4129057982 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TS
2	0.046770	174.143.213.184	192.168.1.2	TCP	74	80 → 54841 [SYN, ACK] Seq=4200111240 Ack=4129057983 Win=5792 Len=0 MSS
3	0.046803	192.168.1.2	174.143.213.184	TCP	66	54841 → 80 [ACK] Seq=4129057983 Ack=4200111241 Win=5888 Len=0 TSval=86
4	0.046841	192.168.1.2	174.143.213.184	HTTP	791	GET /images/layout/logo.png HTTP/1.1

▶ Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
▶ Ethernet II, Src: AsustekC_b3:01:84 (08:1d:60:b3:01:84), Dst: Actionte_2f:47:87 (00:26:62:2f:47:87)
▶ Internet Protocol Version 4, Src: 192.168.1.2, Dst: 174.143.213.184
▶ Transmission Control Protocol, Src Port: 54841, Dst Port: 80, Seq: 4129057982, Len: 0
Source Port: 54841
Destination Port: 80
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 4129057982
Acknowledgment number: 0
Header Length: 40 bytes
▶ Flags: 0x002 (SYN)
Window size value: 5840
[Calculated window size: 5840]
Checksum: 0x85f0 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
▶ Options: (20 bytes), Maximum segment size, SACK permitted, Time

▶ Frame 2: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
▶ Ethernet II, Src: Actionte_2f:47:87 (00:26:62:2f:47:87), Dst: AsustekC_b3:01:84 (08:1d:60:b3:01:84)
▶ Internet Protocol Version 4, Src: 174.143.213.184, Dst: 192.168.1.2
▶ Transmission Control Protocol, Src Port: 80, Dst Port: 54841, Seq: 4200111240, Ack: 4129057983, Len: 0
Source Port: 80
Destination Port: 54841
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 4200111240
Acknowledgment number: 4129057983
Header Length: 40 bytes
▶ Flags: 0x012 (SYN, ACK)
Window size value: 5792
[Calculated window size: 5792]
Checksum: 0x4ff1 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
▶ Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
▶ [SEQ/ACK analysis]

OUTLINE

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

Consider the *3-way_handshake.pcap* file found in this module's resources. Let's now see, how the 3-way handshake looks like on the wire (Wireshark).

1	0.000000	192.168.1.2	174.143.213.184	TCP	74	54841 → 80 [SYN] Seq=4129057982 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TS
2	0.046770	174.143.213.184	192.168.1.2	TCP	74	80 → 54841 [SYN, ACK] Seq=4200111240 Ack=4129057983 Win=5792 Len=0 MSS
3	0.046803	192.168.1.2	174.143.213.184	TCP	66	54841 → 80 [ACK] Seq=4129057983 Ack=4200111241 Win=5888 Len=0 TSval=86
4	0.046841	192.168.1.2	174.143.213.184	HTTP	791	GET /images/layout/logo.png HTTP/1.1

```
Frame 2: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on 0
Ethernet II, Src: Actionte_2f:47:87 (08:26:02:2f:47:87), Dst: AsustekC_b3:81:84 (00:1d:00:b3:81:84)
Internet Protocol Version 4, Src: 174.143.213.184, Dst: 192.168.1.2
Transmission Control Protocol, Src Port: 80, Dst Port: 54841, Seq: 4129057982, Len: 0
  Source Port: 80
  Destination Port: 54841
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence number: 4129057982
  Acknowledgment number: 4129057983
  Header Length: 48 bytes
  Flags: 0x012 (SYN, ACK)
  Window size value: 5792
  [Calculated window size: 5792]
  Checksum: 0x4ff1 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
    Maximum segment size: 1460 bytes
    TCP SACK Permitted Option: True
    Timestamps: TSval 315395185, TSecr 863195
    No-Operation (NOP)
    Window scale: 0 (multiply by 64)
  [SEQ/ACK analysis]
    [This is an ACK to the segment in frame: 1]
    [The RTT to ACK the segment was: 0.046770000 seconds]
    [RTT: 0.046803000 seconds]
```

```
Frame 3: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on 0
Ethernet II, Src: AsustekC_b3:81:84 (00:1d:00:b3:81:84), Dst: Actionte_2f:47:87 (08:26:02:2f:47:87)
Internet Protocol Version 4, Src: 192.168.1.2, Dst: 174.143.213.184
Transmission Control Protocol, Src Port: 54841, Dst Port: 80, Seq: 4129057983, Ack: 4200111241, Len: 0
  Source Port: 54841
  Destination Port: 80
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence number: 4129057983
  Acknowledgment number: 4200111241
  Header Length: 32 bytes
  Flags: 0x010 (ACK)
  Window size value: 46
  [Calculated window size: 5888]
  [Window size scaling factor: 128]
  Checksum: 0x9529 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    No-Operation (NOP)
    No-Operation (NOP)
    Timestamps: TSval 863200, TSecr 315395185
  [SEQ/ACK analysis]
    [This is an ACK to the segment in frame: 2]
    [The RTT to ACK the segment was: 0.000033000 seconds]
    [RTT: 0.046803000 seconds]
```

OUTLINE

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

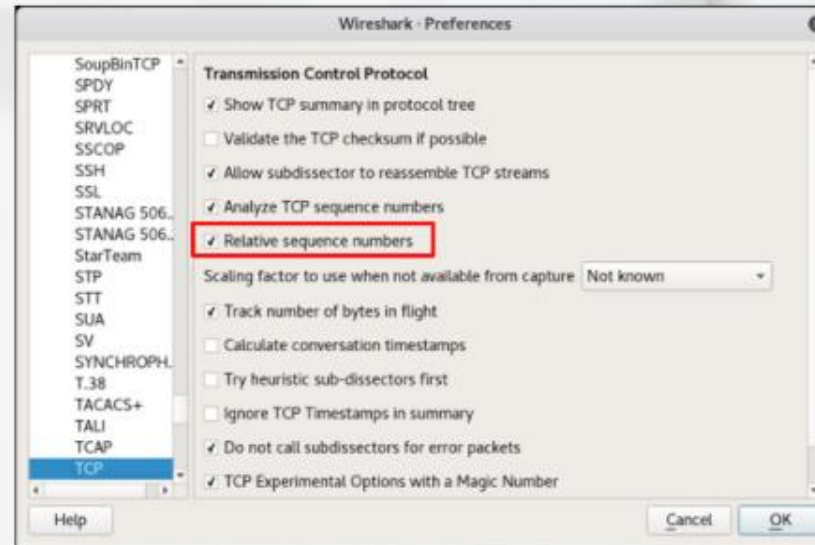
2.1.1 TCP

2.1.1 TCP

Note: Wireshark has an option called *Relative sequence numbers* on, by default. According to the official definition from the [online Wireshark Wiki](https://wiki.wireshark.org/TCP_Relative_Sequence_Numbers),

“By default Wireshark and TShark will keep track of all TCP sessions and convert all Sequence Numbers (SEQ numbers) and Acknowledge Numbers (ACK Numbers) into relative numbers.

This means that instead of displaying the real/absolute SEQ and ACK numbers in the display, Wireshark will display a SEQ and ACK number relative to the first seen segment for that conversation.”



OUTLINE

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

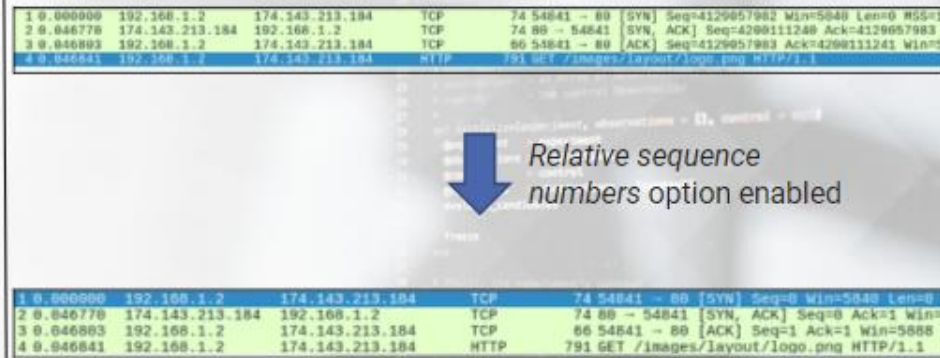
2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

For example, the initial relative sequence number shown in packet #1 (of the *3-way_handshake.pcap* file) is 0. If you look at the ASCII decode though, you'll notice that the actual sequence number is 0xf61c6cbe, or 4129057982 decimal.



1	0.000000	192.168.1.2	174.143.213.184	TCP	74 54841 → 80 [SYN] Seq=4129057982 Win=5840 Len=0 MSS=1
2	0.046778	174.143.213.184	192.168.1.2	TCP	74 80 → 54841 [SYN, ACK] Seq=4209111240 Ack=4129057983
3	0.046893	192.168.1.2	174.143.213.184	TCP	80 54841 → 80 [ACK] Seq=4129057983 Ack=4209111241 Win=0
4	0.046841	192.168.1.2	174.143.213.184	HTTP	791 GET /images/layout/logo.png HTTP/1.1

Relative sequence numbers option enabled



1	0.000000	192.168.1.2	174.143.213.184	TCP	74 54841 → 80 [SYN] Seq=0 Win=5840 Len=0
2	0.046778	174.143.213.184	192.168.1.2	TCP	74 80 → 54841 [SYN, ACK] Seq=0 Ack=1 Win=
3	0.046893	192.168.1.2	174.143.213.184	TCP	80 54841 → 80 [ACK] Seq=1 Ack=1 Win=5888
4	0.046841	192.168.1.2	174.143.213.184	HTTP	791 GET /images/layout/logo.png HTTP/1.1

OUTLINE

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

▼ 2.1.1 TCP

2.1.1.1 Suspicious TCP Traffic

Let's now go through some suspicious TCP traffic cases, but before doing so, find below some tips on what kind of TCP traffic should alarm you.

Normal TCP Traffic	Suspicious TCP Traffic
3 way handshake (SYN, SYN/ACK, ACK)	Excessive SYN packets (scanning)
	Usage of different flags
	Single host to multiple ports or single host to multiple nodes (scanning)

OUTLINE

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

▼ 2.1.1 TCP

2.1.1.1 Suspicious TCP
Traffic

2.1.1.2 Sequence Number Prediction & SYN Scanning

One of the ways using which the venerable Nmap tool tries to perform OS fingerprinting, is by examining the Initial Sequence Numbers (ISNs) generated by the target host (after connections are being attempted to a listening port). Each TCP/IP stack (and subsequently each OS) features its own way of generating Initial Sequence Numbers.

OUTLINE

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

▼ 2.1.1 TCP

2.1.1.1 Suspicious TCP
Traffic

▼ 2.1.1.2 Sequence
Number Prediction & ...

2.1.1.2 Sequence Number Prediction & SYN Scanning

More specifically, Nmap sends a series of SYN packets to an open port and tries to perform OS fingerprinting, by attempting to identify the ISN generation formula in the returned SYN/ACK and comparing the returned values to a file that contains expected returned values for a series of checks.

OUTLINE

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

▼ 2.1.1 TCP

2.1.1.1 Suspicious TCP
Traffic

▼ 2.1.1.2 Sequence
Number Prediction & ...

2.1.1.2 Sequence
Number Predicti...

2.1.1.2 Sequence Number Prediction & SYN Scanning

Open the *nmap_sequence_number_prediction.pcap* file found in this module's resources, to see what could be a Nmap OS fingerprinting attempt.

```
10 2018-01-01 12:00:00.000000000 192.168.1.100 → 192.168.1.1 [RST] Seq=1234567890 Win=0 Len=0
11 2018-01-01 12:00:00.000000000 192.168.1.100 → 192.168.1.1 [RST] Seq=1234567890 Win=0 Len=0
12 2018-01-01 12:00:00.000000000 192.168.1.100 → 192.168.1.1 [RST] Seq=1234567890 Win=0 Len=0
13 2018-01-01 12:00:00.000000000 192.168.1.100 → 192.168.1.1 [RST] Seq=1234567890 Win=0 Len=0
14 2018-01-01 12:00:00.000000000 192.168.1.100 → 192.168.1.1 [RST] Seq=1234567890 Win=0 Len=0
15 2018-01-01 12:00:00.000000000 192.168.1.100 → 192.168.1.1 [RST] Seq=1234567890 Win=0 Len=0
16 2018-01-01 12:00:00.000000000 192.168.1.100 → 192.168.1.1 [RST] Seq=1234567890 Win=0 Len=0
17 2018-01-01 12:00:00.000000000 192.168.1.100 → 192.168.1.1 [RST] Seq=1234567890 Win=0 Len=0
18 2018-01-01 12:00:00.000000000 192.168.1.100 → 192.168.1.1 [RST] Seq=1234567890 Win=0 Len=0
19 2018-01-01 12:00:00.000000000 192.168.1.100 → 192.168.1.1 [RST] Seq=1234567890 Win=0 Len=0
20 2018-01-01 12:00:00.000000000 192.168.1.100 → 192.168.1.1 [RST] Seq=1234567890 Win=0 Len=0
```

OUTLINE

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

▼ 2.1.1 TCP

2.1.1.1 Suspicious TCP
Traffic

▼ 2.1.1.2 Sequence
Number Prediction & ...

2.1.1.2 Sequence
Number Predicti...

2.1.1.2 Sequence
Number Predicti...

2.1.1.2 Sequence Number Prediction & SYN Scanning

Detection

The first thing you should notice, is the big amount of SYN packets being sent to the 192.168.1.2 host (in a short amount of time and without the corresponding SYN/ACK packets).

If you also carefully analyze the series of SYN packets that Nmap sent, you will notice something weird. Nmap repeatedly used the ISN from the scanning host, while scanning the different ports of the remote host. Unique ISNs should be used, while attempting to connect to different ports of a remote host.

1	0.000000	192.168.1.6	192.168.1.2	TCP	58 36901 → 3389 [SYN] Seq=3917476875 Win=1024 Len=0 MSS=1460
2	0.000349	192.168.1.2	192.168.1.6	TCP	60 3389 → 36901 [SYN, ACK] Seq=2694453486 Ack=3917476876 Win=64000 Len=...
3	0.000366	192.168.1.6	192.168.1.2	TCP	58 36901 → 25 [SYN] Seq=3917476875 Win=1024 Len=0 MSS=1460
4	0.000432	192.168.1.6	192.168.1.2	TCP	54 36901 → 3389 [RST] Seq=3917476876 Win=0 Len=0
5	0.000671	192.168.1.6	192.168.1.2	TCP	58 36901 → 135 [SYN] Seq=3917476875 Win=1024 Len=0 MSS=1460
6	0.000906	192.168.1.6	192.168.1.2	TCP	58 36901 → 3306 [SYN] Seq=3917476875 Win=1024 Len=0 MSS=1460
7	0.000973	192.168.1.6	192.168.1.2	TCP	58 36901 → 139 [SYN] Seq=3917476875 Win=1024 Len=0 MSS=1460
8	0.001115	192.168.1.6	192.168.1.2	TCP	58 36901 → 111 [SYN] Seq=3917476875 Win=1024 Len=0 MSS=1460
9	0.001821	192.168.1.6	192.168.1.2	TCP	58 36901 → 143 [SYN] Seq=3917476875 Win=1024 Len=0 MSS=1460
10	0.001825	192.168.1.2	192.168.1.6	TCP	60 135 → 36901 [SYN, ACK] Seq=1710440461 Ack=3917476876 Win=8192 Len=0 ..

OUTLINE

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

▼ 2.1.1 TCP

2.1.1.1 Suspicious TCP Traffic

▼ 2.1.1.2 Sequence Number Prediction & ...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Prediction & SYN Scanning

Detection

Once you identify a scanning host, you can determine if you are dealing with Nmap, by noting the scanning host's ISN and executing the below.

```
>> tcpdump -S -r nmap_sequence_number_prediction.pcap tcp[4:4] = 3917476875
```

The scanning host's ISN reuse, indicates that the scanning/fingerprinting attempt most probably originates from a Nmap instance

```
03:23:10.142073 IP kali.36901 > desktop-agckn5j.3389: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:10.142439 IP kali.36901 > desktop-agckn5j.smtp: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:10.142744 IP kali.36901 > desktop-agckn5j.loc-srv: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:10.142979 IP kali.36901 > desktop-agckn5j.mysql: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:10.143046 IP kali.36901 > desktop-agckn5j.netbios-ssn: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:10.143188 IP kali.36901 > desktop-agckn5j.sunrpc: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:10.143894 IP kali.36901 > desktop-agckn5j.inmap2: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:10.144370 IP kali.36901 > desktop-agckn5j.1723: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:10.144477 IP kali.36901 > desktop-agckn5j.https: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:10.144540 IP kali.36901 > desktop-agckn5j.inaps: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:10.147462 IP kali.36901 > desktop-agckn5j.rtpsp: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:10.147575 IP kali.36901 > desktop-agckn5j.1720: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:10.147638 IP kali.36901 > desktop-agckn5j.http-alt: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:10.147736 IP kali.36901 > desktop-agckn5j.ftp: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:10.147803 IP kali.36901 > desktop-agckn5j.5900: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:10.147903 IP kali.36901 > desktop-agckn5j.snuux: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:11.347428 IP kali.36901 > desktop-agckn5j.microsoft-ds: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:11.347752 IP kali.36901 > desktop-agckn5j.domain: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:11.347864 IP kali.36901 > desktop-agckn5j.ssh: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:11.348057 IP kali.36901 > desktop-agckn5j.pop3s: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:11.348149 IP kali.36901 > desktop-agckn5j.telnet: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:11.348299 IP kali.36901 > desktop-agckn5j.pop3: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
03:23:11.348404 IP kali.36901 > desktop-agckn5j.1025: Flags [S], seq 3917476875, win 1024, options [mss 1460], length 0
```

OUTLINE

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

▼ 2.1.1 TCP

2.1.1.1 Suspicious TCP Traffic

▼ 2.1.1.2 Sequence Number Prediction & ...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

2.1.1.3 Source Port Abnormalities

When it comes to ports, what you should know is that a distinction exists between Privileged (server) ports and Unprivileged (client)/ephemeral ports. Specifically,

- Privileged (server) ports [1-1023] ← Should remain unchanged during the entire connection
- Unprivileged (client)/ephemeral ports [> 1023] ← Chosen only for one connection. Can be chosen again after the connection closes.

OUTLINE

2.1.1 TCP

2.1.1 TCP

2.1.1 TCP

▼ 2.1.1 TCP

2.1.1.1 Suspicious TCP Traffic

▼ 2.1.1.2 Sequence Number Prediction & ...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

▼ 2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

Consider the `nmap_sequence_number_prediction.pcap` file found in this module's resources once again.

Can you spot anything irregular?

OUTLINE

2.1.1 TCP

2.1.1 TCP

▼ 2.1.1 TCP

2.1.1.1 Suspicious TCP Traffic

- 2.1.1.2 Sequence
- ▼ Number Prediction & ...

2.1.1.2 Sequence Number Prediction

2.1.1.2 Sequence
Number Predicti...

2.1.1.2 Sequence Number Prediction

2.1.1.2 Sequence Number Predicti...

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

Detection

ip.src==192.168.1.6						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.6	192.168.1.2	TCP	58	36901 → 3389 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3	0.000366	192.168.1.6	192.168.1.2	TCP	58	36901 → 25 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
4	0.000432	192.168.1.6	192.168.1.2	TCP	54	36901 → 3389 [RST] Seq=1 Win=0 Len=0
5	0.000671	192.168.1.6	192.168.1.2	TCP	58	36901 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
6	0.000906	192.168.1.6	192.168.1.2	TCP	58	36901 → 3306 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
7	0.000973	192.168.1.6	192.168.1.2	TCP	58	36901 → 139 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
8	0.001115	192.168.1.6	192.168.1.2	TCP	58	36901 → 111 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
9	0.001821	192.168.1.6	192.168.1.2	TCP	58	36901 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
12	0.002144	192.168.1.6	192.168.1.2	TCP	54	36901 → 135 [RST] Seq=1 Win=0 Len=0
13	0.002149	192.168.1.6	192.168.1.2	TCP	54	36901 → 139 [RST] Seq=1 Win=0 Len=0

OUTLINE

2.1.1 TCP

▼ 2.1.1 TCP

2.1.1.1 Suspicious TCP Traffic

▼ 2.1.1.2 Sequence Number Prediction & ...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

▼ 2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

Detection

ip.src==192.168.1.6						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.6	192.168.1.2	TCP	58	36901 → 3389 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3	0.000366	192.168.1.6	192.168.1.2	TCP	58	36901 → 25 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
4	0.000432	192.168.1.6	192.168.1.2	TCP	54	36901 → 3389 [RST] Seq=1 Win=0 Len=0
5	0.000671	192.168.1.6	192.168.1.2	TCP	58	36901 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
6	0.000906	192.168.1.6	192.168.1.2	TCP	58	36901 → 3306 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
7	0.000973	192.168.1.6	192.168.1.2	TCP	58	36901 → 139 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
8	0.001115	192.168.1.6	192.168.1.2	TCP	58	36901 → 111 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
9	0.001821	192.168.1.6	192.168.1.2	TCP	58	36901 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
12	0.002144	192.168.1.6	192.168.1.2	TCP	54	36901 → 135 [RST] Seq=1 Win=0 Len=0
13	0.002149	192.168.1.6	192.168.1.2	TCP	54	36901 → 139 [RST] Seq=1 Win=0 Len=0

If you carefully look at packets #1, #3-#9 and #12-#13, you will notice that the host 192.168.1.6 uses the same source port (36901) for multiple connection attempts to different ports of the remote host. This is abnormal. The normal behavior would be incrementing the source port numbers inside the ephemeral port range.

OUTLINE

▼ 2.1.1 TCP

2.1.1.1 Suspicious TCP Traffic

▼ 2.1.1.2 Sequence Number Prediction & ...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

▼ 2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.4 Destination Port Abnormalities

Consider the *strange_scanning.pcap* file found in this module's resources.

Can you spot anything suspicious?

```
14 # ...
15 # ...
16 # ...
17 # ...
18 # ...
19 # ...
20 # ...
21 # ...
22 # ...
23 # ...
24 # ...
25 # ...
26 # ...
27 # ...
28 # ...
29 # ...
30 # ...
31 # ...
32 # ...
33 # ...
34 # ...
35 # ...
36 # ...
37 # ...
38 # ...
39 # ...
40 # ...
41 # ...
42 # ...
43 # ...
44 # ...
45 # ...
46 # ...
47 # ...
48 # ...
49 # ...
50 # ...
51 # ...
52 # ...
53 # ...
54 # ...
55 # ...
56 # ...
57 # ...
58 # ...
59 # ...
60 # ...
61 # ...
62 # ...
63 # ...
64 # ...
65 # ...
66 # ...
67 # ...
68 # ...
69 # ...
70 # ...
71 # ...
72 # ...
73 # ...
74 # ...
75 # ...
76 # ...
77 # ...
78 # ...
79 # ...
80 # ...
81 # ...
82 # ...
83 # ...
84 # ...
85 # ...
86 # ...
87 # ...
88 # ...
89 # ...
90 # ...
91 # ...
92 # ...
93 # ...
94 # ...
95 # ...
96 # ...
97 # ...
98 # ...
99 # ...
100 # ...
```

OUTLINE

2.1.1.1 Suspicious TCP Traffic

▼ 2.1.1.2 Sequence Number Prediction & ...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

▼ 2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

▼ 2.1.1.4 Destination Port Abnormalities

2.1.1.4 Destination Port Abnormalities

Detection

1	0.000000	192.168.1.6	192.168.1.4	TCP	60 2704 → 0 [SYN] Seq=0 Win=512 Len=0
2	0.000036	192.168.1.4	192.168.1.6	TCP	54 0 → 2704 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3	1.000539	192.168.1.6	192.168.1.4	TCP	60 2705 → 0 [SYN] Seq=0 Win=512 Len=0
4	1.000578	192.168.1.4	192.168.1.6	TCP	54 0 → 2705 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
5	2.000999	192.168.1.6	192.168.1.4	TCP	60 2706 → 0 [SYN] Seq=0 Win=512 Len=0
6	2.001037	192.168.1.4	192.168.1.6	TCP	54 0 → 2706 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
7	3.002394	192.168.1.6	192.168.1.4	TCP	60 2707 → 0 [SYN] Seq=0 Win=512 Len=0
8	3.002428	192.168.1.4	192.168.1.6	TCP	54 0 → 2707 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
9	4.003788	192.168.1.6	192.168.1.4	TCP	60 2708 → 0 [SYN] Seq=0 Win=512 Len=0
10	4.003819	192.168.1.4	192.168.1.6	TCP	54 0 → 2708 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
11	5.004594	192.168.1.6	192.168.1.4	TCP	60 2709 → 0 [SYN] Seq=0 Win=512 Len=0
12	5.004632	192.168.1.4	192.168.1.6	TCP	54 0 → 2709 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
13	6.005871	192.168.1.6	192.168.1.4	TCP	60 2710 → 0 [SYN] Seq=0 Win=512 Len=0
14	6.005912	192.168.1.4	192.168.1.6	TCP	54 0 → 2710 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15	7.006765	192.168.1.6	192.168.1.4	TCP	60 2711 → 0 [SYN] Seq=0 Win=512 Len=0
16	7.006798	192.168.1.4	192.168.1.6	TCP	54 0 → 2711 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
17	8.007188	192.168.1.6	192.168.1.4	TCP	60 2712 → 0 [SYN] Seq=0 Win=512 Len=0
18	8.007222	192.168.1.4	192.168.1.6	TCP	54 0 → 2712 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

OUTLINE

2.1.1.2 Sequence Number Prediction & ...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.4 Destination Port Abnormalities

2.1.1.4 Destination Port Abnormalities

2.1.1.4 Destination Port Abnormalities

Detection

1	0.000000	192.168.1.6	192.168.1.4	TCP	60 2704 → 0 [SYN] Seq=0 Win=512 Len=0
2	0.000036	192.168.1.4	192.168.1.6	TCP	54 0 → 2704 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3	1.000539	192.168.1.6	192.168.1.4	TCP	60 2705 → 0 [SYN] Seq=0 Win=512 Len=0
4	1.000578	192.168.1.4	192.168.1.6	TCP	54 0 → 2705 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
5	2.000999	192.168.1.6	192.168.1.4	TCP	60 2706 → 0 [SYN] Seq=0 Win=512 Len=0
6	2.001037	192.168.1.4	192.168.1.6	TCP	54 0 → 2706 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
7	3.002394	192.168.1.6	192.168.1.4	TCP	60 2707 → 0 [SYN] Seq=0 Win=512 Len=0
8	3.002428	192.168.1.4	192.168.1.6	TCP	54 0 → 2707 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
9	4.003788	192.168.1.6	192.168.1.4	TCP	60 2708 → 0 [SYN] Seq=0 Win=512 Len=0
10	4.003819	192.168.1.4	192.168.1.6	TCP	54 0 → 2708 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
11	5.004594	192.168.1.6	192.168.1.4	TCP	60 2709 → 0 [SYN] Seq=0 Win=512 Len=0
12	5.004632	192.168.1.4	192.168.1.6	TCP	54 0 → 2709 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
13	6.005871	192.168.1.6	192.168.1.4	TCP	60 2710 → 0 [SYN] Seq=0 Win=512 Len=0
14	6.005912	192.168.1.4	192.168.1.6	TCP	54 0 → 2710 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15	7.006765	192.168.1.6	192.168.1.4	TCP	60 2711 → 0 [SYN] Seq=0 Win=512 Len=0
16	7.006798	192.168.1.4	192.168.1.6	TCP	54 0 → 2711 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
17	8.007188	192.168.1.6	192.168.1.4	TCP	60 2712 → 0 [SYN] Seq=0 Win=512 Len=0
18	8.007222	192.168.1.4	192.168.1.6	TCP	54 0 → 2712 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Obviously, we are dealing with a scanning attempt, but what is that destination port 0 (TCP) all about? A host is not supposed to be listening on port 0. A scanning tool sending packets to port 0 (TCP) of the remote host, is trying identify if the remote host is alive. If the remote host is alive, then, a **RST** response should be sent to all those packets.

OUTLINE

2.1.1.1 Sequence Number Prediction

2.1.1.2 Sequence Number Prediction

2.1.1.2 Sequence Number Prediction

2.1.1.2 Sequence Number Prediction

2.1.1.2 Sequence Number Prediction

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.4 Destination Port Abnormalities

2.1.1.4 Destination Port Abnormalities

2.1.1.4 Destination Port Abnormalities

2.1.1.5 TCP RST Attack

It has been known for years that an existing TCP connection can be cut apart through a crafted TCP RST (and other) packet sent either to the client or the server. For a successful RST attack to be executed, an attacker should have prior knowledge of the below.

- **Source Port**
- **Destination Port**
- **Source IP**
- **Destination IP**
- **“correct” Sequence Number**

```
18 # detect the source port of the connection
19 # detect the destination port of the connection
20 # detect the source IP of the connection
21 # detect the destination IP of the connection
22 # detect the sequence number of the connection
23 # detect the window size of the connection
24 # detect the window scale of the connection
25 # detect the window size of the connection
26 # detect the window scale of the connection
27 # detect the window size of the connection
28 # detect the window scale of the connection
29 # detect the window size of the connection
30 # detect the window scale of the connection
31 # detect the window size of the connection
32 # detect the window scale of the connection
33 # detect the window size of the connection
34 # detect the window scale of the connection
35 # detect the window size of the connection
36 # detect the window scale of the connection
37 # detect the window size of the connection
38 # detect the window scale of the connection
39 # detect the window size of the connection
40 # detect the window scale of the connection
41 # detect the window size of the connection
42 # detect the window scale of the connection
43 # detect the window size of the connection
44 # detect the window scale of the connection
45 # detect the window size of the connection
46 # detect the window scale of the connection
47 # detect the window size of the connection
48 # detect the window scale of the connection
49 # detect the window size of the connection
50 # detect the window scale of the connection
```

OUTLINE

2.1.1.1 Source Port Abnormalities

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

▼ 2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

▼ 2.1.1.4 Destination Port Abnormalities

2.1.1.4 Destination Port Abnormalities

2.1.1.4 Destination Port Abnormalities

▼ 2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

It has been known for years that an existing TCP connection can be cut apart through a crafted TCP RST (and other) packet sent either to the client or the server. For a successful RST attack to be executed, an attacker should have prior knowledge of the below.

- **Source Port**
- **Destination Port**
- **Source IP**
- **Destination IP**
- **“correct” Sequence Number**

Those can be easily identified by an attacker in the same network, who is sniffing the network traffic.

TCP RST attacks can also be executed remotely though, as described in the following paper
https://packetstormsecurity.com/files/download/33170/SlippingInTheWindow_v1.0.doc

OUTLINE

2.1.1.1 Introduction

2.1.1.2 Sequence Number Predicti...

2.1.1.2 Sequence Number Predicti...

▼ 2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

▼ 2.1.1.4 Destination Port Abnormalities

2.1.1.4 Destination Port Abnormalities

2.1.1.4 Destination Port Abnormalities

▼ 2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

Consider the *TCP_RST_Attack.pcap* file found in this module's resources.

Try to identify any abnormalities.

```
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2203  
2204  
2205  
2206  
2207  
2208  
2209  
2210  
2211  
2212  
2213  
2214  
2215  
2216  
2217  
2218  
2219  
2220  
2221  
2222  
2223  
2224  
2225  
2226  
2227  
2228  
2229  
2230  
2231
```

2.1.1.5 TCP RST Attack

Detection

1 0.000000 192.168.1.4 192.168.1.5 TCP	74 34326 → 4444 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=348934
2 0.000474 192.168.1.5 192.168.1.4 TCP	74 4444 → 34326 [SYN, ACK] Seq=0 Ack=1 Win=20960 Len=0 MSS=1460 SACK_PERM=1 T
3 0.000503 192.168.1.4 192.168.1.5 TCP	66 34326 → 4444 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3489344369 TSecr=2162
4 0.046034 192.168.1.4 192.168.1.5 TCP	85 34326 → 4444 [PSH, ACK] Seq=1 Ack=1 Win=29312 Len=19 TSval=3489353414 TSec
5 0.046411 192.168.1.5 192.168.1.4 TCP	66 4444 → 34326 [ACK] Seq=1 Ack=20 Win=29056 Len=0 TSval=2162012242 TSecr=348
6 26.230271 192.168.1.5 192.168.1.4 TCP	96 4444 → 34326 [PSH, ACK] Seq=1 Ack=20 Win=29056 Len=30 TSval=2162029426 TSe
7 26.230302 192.168.1.4 192.168.1.5 TCP	66 34326 → 4444 [ACK] Seq=20 Ack=31 Win=29312 Len=0 TSval=3489370598 TSecr=21
8 26.721820 192.168.1.2 78.108.120.24 SSL	60 Continuation Data
9 26.780548 78.108.120.24 192.168.1.2 TCP	60 443 → 49799 [ACK] Seq=1 Ack=2 Win=40958 Len=0
10 122.5011 192.168.1.4 192.168.1.5 TCP	99 34326 → 4444 [PSH, ACK] Seq=20 Ack=31 Win=29312 Len=33 TSval=3489466928 TS
11 122.5016 192.168.1.5 192.168.1.4 TCP	66 4444 → 34326 [ACK] Seq=31 Ack=53 Win=29056 Len=0 TSval=2162125756 TSecr=34
12 122.6207 192.168.1.5 192.168.1.4 TCP	66 4444 → 34326 [RST, ACK] Seq=31 Ack=21 Win=0 Len=0
13 122.6207 192.168.1.4 192.168.1.5 TCP	66 [TCP ACKed unseen segment] 34326 → 4444 [RST, ACK] Seq=53 Ack=32 Win=0 Len

• Frame 11: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
• Ethernet II, Src: Vmware_48:d5:8d (00:0c:29:46:d5:8d), Dst: Vmware_a3:eb:77 (00:0c:29:a3:eb:77)
• Internet Protocol Version 4, Src: 192.168.1.5, Dst: 192.168.1.4
• Transmission Control Protocol, Src Port: 4444, Dst Port: 34326, Seq: 31, Ack: 53, Len: 0

1 0.000000 192.168.1.4 192.168.1.5 TCP	74 34326 → 4444 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=348934
2 0.000474 192.168.1.5 192.168.1.4 TCP	74 4444 → 34326 [SYN, ACK] Seq=0 Ack=1 Win=20960 Len=0 MSS=1460 SACK_PERM=1 T
3 0.000503 192.168.1.4 192.168.1.5 TCP	66 34326 → 4444 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3489344369 TSecr=2162
4 0.046034 192.168.1.4 192.168.1.5 TCP	85 34326 → 4444 [PSH, ACK] Seq=1 Ack=1 Win=29312 Len=19 TSval=3489353414 TSec
5 0.046411 192.168.1.5 192.168.1.4 TCP	66 4444 → 34326 [ACK] Seq=1 Ack=20 Win=29056 Len=0 TSval=2162012242 TSecr=348
6 26.230271 192.168.1.5 192.168.1.4 TCP	96 4444 → 34326 [PSH, ACK] Seq=1 Ack=20 Win=29056 Len=30 TSval=2162029426 TSe
7 26.230302 192.168.1.4 192.168.1.5 TCP	66 34326 → 4444 [ACK] Seq=20 Ack=31 Win=29312 Len=0 TSval=3489370598 TSecr=21
8 26.721820 192.168.1.2 78.108.120.24 SSL	60 Continuation Data
9 26.780548 78.108.120.24 192.168.1.2 TCP	60 443 → 49799 [ACK] Seq=1 Ack=2 Win=40958 Len=0
10 122.5011 192.168.1.4 192.168.1.5 TCP	99 34326 → 4444 [PSH, ACK] Seq=20 Ack=31 Win=29312 Len=33 TSval=3489466928 TS
11 122.5016 192.168.1.5 192.168.1.4 TCP	66 4444 → 34326 [ACK] Seq=31 Ack=53 Win=29056 Len=0 TSval=2162125756 TSecr=34
12 122.6207 192.168.1.5 192.168.1.4 TCP	66 4444 → 34326 [RST, ACK] Seq=31 Ack=21 Win=0 Len=0
13 122.6207 192.168.1.4 192.168.1.5 TCP	66 [TCP ACKed unseen segment] 34326 → 4444 [RST, ACK] Seq=53 Ack=32 Win=0 Len

• Frame 12: 66 bytes on wire (480 bits), 66 bytes captured (480 bits)
• Ethernet II, Src: Vmware_d5:f7:aa (00:0c:29:d5:f7:aa), Dst: Vmware_a3:eb:77 (00:0c:29:a3:eb:77)
• Internet Protocol Version 4, Src: 192.168.1.5, Dst: 192.168.1.4
• Transmission Control Protocol, Src Port: 4444, Dst Port: 34326, Seq: 31, Ack: 21, Len: 0

OUTLINE

- 2.1.1.3 Source Port Abnormalities
 - 2.1.1.3 Source Port Abnormalities
 - 2.1.1.3 Source Port Abnormalities
 - 2.1.1.3 Source Port Abnormalities
- 2.1.1.4 Destination Port Abnormalities
 - 2.1.1.4 Destination Port Abnormalities
 - 2.1.1.4 Destination Port Abnormalities
- 2.1.1.5 TCP RST Attack
 - 2.1.1.5 TCP RST Attack
 - 2.1.1.5 TCP RST Attack
 - 2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

Detection

1 0.000000	192.168.1.4	192.168.1.5	TCP	74 34326 - 4444 [SYN] Seq=0 Win=29008 Len=0 MSS=1460 SACK_PERM=1 TSval=348934
2 0.000474	192.168.1.5	192.168.1.4	TCP	74 4444 - 34326 [SYN, ACK] Seq=0 Ack=1 Win=29008 Len=0 MSS=1460 SACK_PERM=1 T...
3 0.000503	192.168.1.4	192.168.1.5	TCP	66 34326 - 4444 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3489344369 TSecr=2182...
4 0.046034	192.168.1.4	192.168.1.5	TCP	85 34326 - 4444 [PSH, ACK] Seq=1 Ack=1 Win=29312 Len=19 TSval=3489353414 TSec...
5 0.046411	192.168.1.5	192.168.1.4	TCP	66 4444 - 34326 [ACK] Seq=1 Ack=20 Win=29056 Len=0 TSval=2182012242 TSecr=348...
6 26.230271	192.168.1.5	192.168.1.4	TCP	96 4444 - 34326 [PSH, ACK] Seq=1 Ack=20 Win=29056 Len=38 TSval=2182029426 TSe...
7 26.230302	192.168.1.4	192.168.1.5	TCP	66 34326 - 4444 [ACK] Seq=20 Ack=31 Win=29312 Len=0 TSval=3489370598 TSecr=21...
8 26.721820	192.168.1.2	78.108.120.24	SSL	60 Continuation Data
9 26.786548	78.108.120.24	192.168.1.2	TCP	66 443 - 49799 [ACK] Seq=1 Ack=2 Win=40958 Len=0
10 122.5611	192.168.1.4	192.168.1.5	TCP	99 34326 - 4444 [PSH, ACK] Seq=20 Ack=31 Win=29312 Len=33 TSval=3489466928 TS...
11 122.5616	192.168.1.5	192.168.1.4	TCP	66 4444 - 34326 [ACK] Seq=31 Ack=53 Win=29056 Len=0 TSval=2182125756 TSecr=34...
12 122.6207	192.168.1.5	192.168.1.4	TCP	66 4444 - 34326 [RST, ACK] Seq=31 Ack=21 Win=0 Len=0
13 122.6207	192.168.1.4	192.168.1.5	TCP	66 [TCP ACKed unseen segment] 34326 - 4444 [RST, ACK] Seq=53 Ack=32 Win=0 Len...
* Frame 12: 66 bytes on wire (480 bits), 66 bytes captured (480 bits)				
* Ethernet II, Src: Vmware_d5:f7:aa (08:0c:29:d5:f7:aa), Dst: Vmware_a3:eb:77 (08:0c:29:a3:eb:77)				
* Internet Protocol Version 4, Src: 192.168.1.5, Dst: 192.168.1.4				
* Transmission Control Protocol, Src Port: 4444, Dst Port: 34326, Seq: 31, Ack: 21, Len: 0				

1 0.000000	192.168.1.4	192.168.1.5	TCP	74 34326 - 4444 [SYN] Seq=0 Win=29008 Len=0 MSS=1460 SACK_PERM=1 TSval=348934
2 0.000474	192.168.1.5	192.168.1.4	TCP	74 4444 - 34326 [SYN, ACK] Seq=0 Ack=1 Win=29008 Len=0 MSS=1460 SACK_PERM=1 T...
3 0.000503	192.168.1.4	192.168.1.5	TCP	66 34326 - 4444 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3489344369 TSecr=2182...
4 0.046034	192.168.1.4	192.168.1.5	TCP	85 34326 - 4444 [PSH, ACK] Seq=1 Ack=1 Win=29312 Len=19 TSval=3489353414 TSec...
5 0.046411	192.168.1.5	192.168.1.4	TCP	66 4444 - 34326 [ACK] Seq=1 Ack=20 Win=29056 Len=0 TSval=2182012242 TSecr=348...
6 26.230271	192.168.1.5	192.168.1.4	TCP	96 4444 - 34326 [PSH, ACK] Seq=1 Ack=20 Win=29056 Len=38 TSval=2182029426 TSe...
7 26.230302	192.168.1.4	192.168.1.5	TCP	66 34326 - 4444 [ACK] Seq=20 Ack=31 Win=29312 Len=0 TSval=3489370598 TSecr=21...
8 26.721820	192.168.1.2	78.108.120.24	SSL	60 Continuation Data
9 26.786548	78.108.120.24	192.168.1.2	TCP	66 443 - 49799 [ACK] Seq=1 Ack=2 Win=40958 Len=0
10 122.5611	192.168.1.4	192.168.1.5	TCP	99 34326 - 4444 [PSH, ACK] Seq=20 Ack=31 Win=29312 Len=33 TSval=3489466928 TS...
11 122.5616	192.168.1.5	192.168.1.4	TCP	66 4444 - 34326 [ACK] Seq=31 Ack=53 Win=29056 Len=0 TSval=2182125756 TSecr=34...
12 122.6207	192.168.1.5	192.168.1.4	TCP	66 4444 - 34326 [RST, ACK] Seq=31 Ack=21 Win=0 Len=0
13 122.6207	192.168.1.4	192.168.1.5	TCP	66 [TCP ACKed unseen segment] 34326 - 4444 [RST, ACK] Seq=53 Ack=32 Win=0 Len...
* Frame 11: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)				
* Ethernet II, Src: Vmware_d6:d5:8d (08:0c:29:d6:d5:8d), Dst: Vmware_a3:eb:77 (08:0c:29:a3:eb:77)				
* Internet Protocol Version 4, Src: 192.168.1.5, Dst: 192.168.1.4				
* Transmission Control Protocol, Src Port: 4444, Dst Port: 34326, Seq: 31, Ack: 53, Len: 0				

- If you carefully look at packet #12, you will notice that the 192.168.1.5 host sent a TCP RST packet and subsequently terminated the connection. So far, everything seems normal. Now, if you carefully look at packet #11, you will notice normal data transmission originating from the 192.168.1.5 host, but its MAC address is different than the one included in packet #12.
- It looks like an attacker executed a TCP RST attack to terminate the connection, spoofing the 192.168.1.5 host.
- The next sequence number was first identified through sniffing traffic and then used inside the spoofed RST packet.

OUTLINE

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

2.1.1.4 Destination Port Abnormalities

2.1.1.4 Destination Port Abnormalities

2.1.1.4 Destination Port Abnormalities

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.6 TCP Session Hijacking

As you might have figured out by now, TCP is defenseless against “packet injection” attacks. An attacker could choose to hijack a whole TCP session, instead of executing a simple TCP RST attack.

TCP session hijacking requires the same prior knowledge as the TCP RST attack.

OUTLINE

2.1.1.1 Introduction

2.1.1.3 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

▼ 2.1.1.4 Destination Port Abnormalities

2.1.1.4 Destination Port Abnormalities

2.1.1.4 Destination Port Abnormalities

▼ 2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

▼ 2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

Consider the *TCP-hijacking.pcap* file found in this module's resources.

Try to identify any abnormalities.

```
24 # ...
25 # ...
26 # ...
27 # ...
28 # ...
29 # ...
30 # ...
31 # ...
32 # ...
33 # ...
34 # ...
35 # ...
36 # ...
37 # ...
38 # ...
39 # ...
40 # ...
41 # ...
42 # ...
43 # ...
44 # ...
45 # ...
46 # ...
47 # ...
48 # ...
49 # ...
50 # ...
51 # ...
52 # ...
53 # ...
54 # ...
55 # ...
56 # ...
57 # ...
58 # ...
59 # ...
60 # ...
61 # ...
62 # ...
63 # ...
64 # ...
65 # ...
66 # ...
67 # ...
68 # ...
69 # ...
70 # ...
71 # ...
72 # ...
73 # ...
74 # ...
75 # ...
76 # ...
77 # ...
78 # ...
79 # ...
80 # ...
81 # ...
82 # ...
83 # ...
84 # ...
85 # ...
86 # ...
87 # ...
88 # ...
89 # ...
90 # ...
91 # ...
92 # ...
93 # ...
94 # ...
95 # ...
96 # ...
97 # ...
98 # ...
99 # ...
100 # ...
```

OUTLINE

2.1.1.1 Source Port Abnormalities

2.1.1.3 Source Port Abnormalities

▼ 2.1.1.4 Destination Port Abnormalities

2.1.1.4 Destination Port Abnormalities

2.1.1.4 Destination Port Abnormalities

▼ 2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

▼ 2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

Detection

1	0.000000	192.168.1.4	192.168.1.5	TELNET	67 Telnet Data ...
2	0.001212	192.168.1.5	192.168.1.4	TELNET	67 Telnet Data ...
3	0.001829	192.168.1.4	192.168.1.5	TCP	66 36212 → 23 [ACK] Seq=3123452945 Ack=1235858240 Win=237 Len=0 TSval=4214429...
4	0.136173	192.168.1.4	192.168.1.5	TELNET	67 Telnet Data ...
5	0.136718	192.168.1.5	192.168.1.4	TELNET	67 Telnet Data ...
6	0.136958	192.168.1.4	192.168.1.5	TCP	66 36212 → 23 [ACK] Seq=3123452946 Ack=1235858241 Win=237 Len=0 TSval=4214429...
7	0.487909	192.168.1.4	192.168.1.5	TELNET	68 Telnet Data ...
8	0.488647	192.168.1.5	192.168.1.4	TELNET	68 Telnet Data ...
9	0.488846	192.168.1.4	192.168.1.5	TCP	66 36212 → 23 [ACK] Seq=3123452948 Ack=1235858243 Win=237 Len=0 TSval=4214429...
10	0.490814	192.168.1.5	192.168.1.4	TELNET	140 Telnet Data ...
11	0.490969	192.168.1.4	192.168.1.5	TCP	66 36212 → 23 [ACK] Seq=3123452948 Ack=1235858317 Win=237 Len=0 TSval=4214429...
12	6.264193	192.168.1.4	192.168.1.5	TELNET	67 Telnet Data ...
13	6.265435	192.168.1.5	192.168.1.4	TELNET	67 Telnet Data ...
14	6.265727	192.168.1.4	192.168.1.5	TCP	66 36212 → 23 [ACK] Seq=3123452949 Ack=1235858318 Win=237 Len=0 TSval=4214435...
15	6.282426	192.168.1.4	192.168.1.5	TCP	1078 [TCP Retransmission] 36212 → 23 [PSH, ACK] Seq=3123452948 Ack=1235858317 W...
16	6.282906	192.168.1.5	192.168.1.4	TCP	78 [TCP ACKed unseen segment] 23 → 36212 [ACK] Seq=1235858318 Ack=3123453972 ...
17	14.488028	192.168.1.4	192.168.1.5	TELNET	63 Telnet Data ...
18	14.491266	192.168.1.5	192.168.1.4	TCP	66 23 → 36212 [ACK] Seq=1235858318 Ack=3123453981 Win=243 Len=0 TSval=2774555...
19	14.491272	192.168.1.5	192.168.1.4	TELNET	76 Telnet Data ...
20	14.696517	192.168.1.5	192.168.1.4	TELNET	230 Telnet Data ...
21	14.905919	192.168.1.5	192.168.1.4	TCP	240 [TCP Retransmission] 23 → 36212 [PSH, ACK] Seq=1235858318 Ack=3123453981 W...
22	15.322954	192.168.1.5	192.168.1.4	TCP	240 [TCP Retransmission] 23 → 36212 [PSH, ACK] Seq=1235858318 Ack=3123453981 W...
23	16.156205	192.168.1.5	192.168.1.4	TCP	240 [TCP Retransmission] 23 → 36212 [PSH, ACK] Seq=1235858318 Ack=3123453981 W...

OUTLINE

2.1.1.4 Destination Port Abnormalities

2.1.1.4 Destination Port Abnormalities

2.1.1.4 Destination Port Abnormalities

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

Detection

1	0.000000	192.168.1.4	192.168.1.5	TELNET	67 Telnet Data ...
2	0.001212	192.168.1.5	192.168.1.4	TELNET	67 Telnet Data ...
3	0.001829	192.168.1.4	192.168.1.5	TCP	66 36212 → 23 [ACK] Seq=3123452945 Ack=1235858240 Win=237 Len=0 TSval=4214429..
4	0.136173	192.168.1.4	192.168.1.5	TELNET	67 Telnet Data ...
5	0.136718	192.168.1.5	192.168.1.4	TELNET	67 Telnet Data ...
6	0.136958	192.168.1.4	192.168.1.5	TCP	66 36212 → 23 [ACK] Seq=3123452946 Ack=1235858241 Win=237 Len=0 TSval=4214429..
7	0.487909	192.168.1.4	192.168.1.5	TELNET	68 Telnet Data ...
8	0.488647	192.168.1.5	192.168.1.4	TELNET	68 Telnet Data ...
9	0.488846	192.168.1.4	192.168.1.5	TCP	66 36212 → 23 [ACK] Seq=3123452948 Ack=1235858243 Win=237 Len=0 TSval=4214429..
10	0.490814	192.168.1.5	192.168.1.4	TELNET	140 Telnet Data ...
11	0.490969	192.168.1.4	192.168.1.5	TCP	66 36212 → 23 [ACK] Seq=3123452948 Ack=1235858317 Win=237 Len=0 TSval=4214429..
12	0.264193	192.168.1.4	192.168.1.5	TELNET	67 Telnet Data ...
13	0.265435	192.168.1.5	192.168.1.4	TELNET	67 Telnet Data ...
14	0.265727	192.168.1.4	192.168.1.5	TCP	66 36212 → 23 [ACK] Seq=3123452949 Ack=1235858318 Win=237 Len=0 TSval=4214435..
15	0.282426	192.168.1.4	192.168.1.5	TCP	1078 [TCP Retransmission] 36212 → 23 [PSH, ACK] Seq=3123452948 Ack=1235858317 W..
16	0.282996	192.168.1.5	192.168.1.4	TCP	78 [TCP ACKed unseen segment] 23 → 36212 (ACK) Seq=1235858318 Ack=3123453972 ..
17	14.488828	192.168.1.4	192.168.1.5	TELNET	63 Telnet Data ...
18	14.491266	192.168.1.5	192.168.1.4	TCP	66 23 → 36212 [ACK] Seq=1235858318 Ack=3123453981 Win=243 Len=0 TSval=2774555..
19	14.491272	192.168.1.5	192.168.1.4	TELNET	76 Telnet Data ...
20	14.696517	192.168.1.5	192.168.1.4	TELNET	230 Telnet Data ...
21	14.905919	192.168.1.5	192.168.1.4	TCP	240 [TCP Retransmission] 23 → 36212 [PSH, ACK] Seq=1235858318 Ack=3123453981 W..
22	15.322954	192.168.1.5	192.168.1.4	TCP	240 [TCP Retransmission] 23 → 36212 [PSH, ACK] Seq=1235858318 Ack=3123453981 W..
23	16.156205	192.168.1.5	192.168.1.4	TCP	240 [TCP Retransmission] 23 → 36212 [PSH, ACK] Seq=1235858318 Ack=3123453981 W..

- This is obviously a capture file of a Telnet session. Telnet offers no encryption and thus, we can see every command the client issued and every result returned by the server in clear text.
- Let's analyze packet #15.
 - TCP Retransmission is displayed because the sequence number and the acknowledgement number of this packet are the same as the ones in packet #11.
 - The MAC address of the client (192.168.1.4) in this packet is different than the MAC address that is included in all previous packets related to this host.
- It looks like an attacker has taken over (hijacked) the whole Telnet session. This is also apparent in packet #17, that includes the MAC address of the attacker and the command the attacker issued (`uname -a`)
- The server has no defense mechanism to detect that the Telnet session is hijacked and sends the output of the `uname -a` command back to the 192.168.1.4 client.

OUTLINE

2.1.1.4 Destination
Port Abnormalities

2.1.1.4 Destination
Port Abnormalities

▼ 2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST
Attack

2.1.1.5 TCP RST
Attack

2.1.1.5 TCP RST
Attack

2.1.1.5 TCP RST
Attack

▼ 2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session
Hijacking

2.1.1.6 TCP Session
Hijacking

2.1.1.6 TCP Session
Hijacking

2.1.1.7 TCP Options Abuse

At the end of the TCP header, one may find TCP options being specified.

Historically, TCP options have been abused for evasion and information gathering purposes.

```
23  # ...
24  # ...
25  # ...
26  # ...
27  # ...
28  # ...
29  # ...
30  # ...
31  # ...
32  # ...
33  # ...
34  # ...
35  # ...
36  # ...
37  # ...
38  # ...
39  # ...
40  # ...
41  # ...
42  # ...
43  # ...
44  # ...
45  # ...
46  # ...
47  # ...
48  # ...
49  # ...
50  # ...
51  # ...
52  # ...
53  # ...
54  # ...
55  # ...
56  # ...
57  # ...
58  # ...
59  # ...
60  # ...
61  # ...
62  # ...
63  # ...
64  # ...
65  # ...
66  # ...
67  # ...
68  # ...
69  # ...
70  # ...
71  # ...
72  # ...
73  # ...
74  # ...
75  # ...
76  # ...
77  # ...
78  # ...
79  # ...
80  # ...
81  # ...
82  # ...
83  # ...
84  # ...
85  # ...
86  # ...
87  # ...
88  # ...
89  # ...
90  # ...
91  # ...
92  # ...
93  # ...
94  # ...
95  # ...
96  # ...
97  # ...
98  # ...
99  # ...
100 # ...
```



OUTLINE

2.1.1.1 TCP Header

2.1.1.4 Destination
Port Abnormalities

▼ 2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST
Attack

2.1.1.5 TCP RST
Attack

2.1.1.5 TCP RST
Attack

2.1.1.5 TCP RST
Attack

▼ 2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session
Hijacking

2.1.1.6 TCP Session
Hijacking

2.1.1.6 TCP Session
Hijacking

▼ 2.1.1.7 TCP Options Abuse

2.1.1.7.1 TCP Timestamps Option

One example of a TCP option that can be abused, is the [TCP Timestamps option](#).

According to RFC 1323, the Timestamps option carries two four-byte timestamp fields. The Timestamp Value field (TSval) contains the current value of the timestamp clock of the TCP sending the option. The Timestamp Echo Reply field (TSecr) is only valid if the ACK bit is set in the TCP header; if it is valid, it “echos” a timestamp value that was sent by the remote TCP in the TSval field of a Timestamps option.

OUTLINE

▼ 2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

▼ 2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

▼ 2.1.1.7 TCP Options Abuse

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.1 TCP Timestamps Option

Specifically, the TCP Timestamps options can be abused in order to:

1. Determine the patch level of a system, through uptime analysis
2. Perform host identification using [clock skew](#)
3. Identify how a target DMZ is structured

OUTLINE

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

▼ 2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

▼ 2.1.1.7 TCP Options Abuse

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.1 TCP Timestamps Option

Technical details on how attackers are abusing the TCP Timestamps option can be found in the following link.

<https://www.scip.ch/en/?labs.20150305>

OUTLINE

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

▼ 2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

▼ 2.1.1.7 TCP Options Abuse

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.2 Leveraging TCP Option Support & Ordering

Oftentimes, TCP/IP stacks, support a subset of the available TCP options and also, perform TCP option storing in their own unique order. Nmap leverages the above (and other things), in order to perform OS fingerprinting.

OUTLINE

2.1.1.7.2 Leveraging TCP Option Support & Ordering

2.1.1.5 TCP RST Attack

2.1.1.5 TCP RST Attack

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

2.1.1.7 TCP Options Abuse

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.2 Leveraging TCP Option Support & Ordering

2.1.1.7.2 Leveraging TCP Option Support & Ordering

The following article offers technical details (as well as an example) on how OS fingerprinting can be achieved by analyzing TCP options (refer to the *TCP Options* section).

<https://nmap.org/nmap-fingerprinting-article.txt>

OUTLINE

2.1.1.5 TCP RST

Attack

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

2.1.1.7 TCP Options Abuse

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.2 Leveraging TCP Option Supp...

2.1.1.7.2 Leveraging TCP Option Supp...

2.1.2 UDP

UDP Overview

- ✓ UDP is a less complicated (than TCP) and lightweight protocol, due to the lack of fields that ensure reliable delivery
- ✓ Unlike TCP, UDP can deliver traffic to multiple hosts.
- ✓ Privileged (server) and ephemeral (client) ports exist, just like in TCP (For each new connection, a different ephemeral port should be utilized)
- ✓ IPv6 dictated some changes such as:
 - UDP length 0 that designates a jumbogram packet
 - The requirement for UDP checksums

OUTLINE

- 2.1.1.6 TCP Session Hijacking
 - 2.1.1.6 TCP Session Hijacking
 - 2.1.1.6 TCP Session Hijacking
 - 2.1.1.6 TCP Session Hijacking
- 2.1.1.7 TCP Options Abuse
 - 2.1.1.7.1 TCP Timestamps Option
 - 2.1.1.7.1 TCP Timestamps Option
 - 2.1.1.7.1 TCP Timestamps Option
 - 2.1.1.7.2 Leveraging TCP Option Supp...
 - 2.1.1.7.2 Leveraging TCP Option Supp...

2.1.2 UDP

UDP-based attacking techniques, such as DNS Command & Control, DNS exfiltration etc. will be covered as the course progresses. Hang on...

OUTLINE

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

▼ 2.1.1.7 TCP Options Abuse

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.2 Leveraging TCP Option Supp...

2.1.1.7.2 Leveraging TCP Option Supp...

▼ 2.1.2 UDP

2.1.2 UDP

2.1.3 ICMP

ICMP Overview

- ✓ The Internet Control Message Protocol (ICMP) is another less complicated (than TCP) and lightweight protocol
- ✓ It is commonly used for network troubleshooting and delivering error messages
- ✓ It is an unreliable protocol
- ✓ It can send and respond to broadcast messages
- ✓ There are no ports and client-server communications involved

OUTLINE

2.1.1.5 ICMP

2.1.1.6 TCP Session Hijacking

2.1.1.6 TCP Session Hijacking

▼ 2.1.1.7 TCP Options Abuse

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.2 Leveraging TCP Option Supp...

2.1.1.7.2 Leveraging TCP Option Supp...

▼ 2.1.2 UDP

2.1.2 UDP

▼ 2.1.3 ICMP

2.1.3.1 ICMP Abuse

Historically, ICMP has been abused for all kinds of nefarious purposes, from mapping and reconnaissance all the way to DDoS and Man-In-The-Middle attacks.

OUTLINE

2.1.1.5 ICMP

2.1.1.6 TCP Session Hijacking

▼ 2.1.1.7 TCP Options Abuse

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.2 Leveraging TCP Option Supp...

2.1.1.7.2 Leveraging TCP Option Supp...

▼ 2.1.2 UDP

2.1.2 UDP

▼ 2.1.3 ICMP

2.1.3.1 ICMP Abuse

2.1.3.1 ICMP Abuse

We all know that we can map live hosts via ICMP echo requests to a single, broadcast or subnet address. This mapping technique though has been abused in the past to such a degree that most sites nowadays disallow inbound and/or outbound ICMP echo requests.

This fact triggered the creation of a new wave of ICMP-based mapping techniques by attackers.

OUTLINE

engines

▼ 2.1.1.7 TCP Options Abuse

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.1 TCP Timestamps Option

2.1.1.7.2 Leveraging TCP Option Supp...

2.1.1.7.2 Leveraging TCP Option Supp...

▼ 2.1.2 UDP

2.1.2 UDP

▼ 2.1.3 ICMP

2.1.3.1 ICMP Abuse

▼ 2.1.3.1 ICMP Abuse

2.1.3.1.1 ICMP Address Mask Request/Reply

ICMP can be used to identify a target host's subnet mask. Such information can be useful in distinguishing subnets within an IP address range.

This can be achieved, by sending an ICMP address mask request to a given target. Such a request may elicit responses from gateway hosts or other network devices **only**.

OUTLINE

2.1.1.7.1 TCP
Timestamps Option

2.1.1.7.1 TCP
Timestamps Option

2.1.1.7.1 TCP
Timestamps Option

2.1.1.7.2 Leveraging
TCP Option Supp...

2.1.1.7.2 Leveraging
TCP Option Supp...

▼ 2.1.2 UDP

2.1.2 UDP

▼ 2.1.3 ICMP

2.1.3.1 ICMP Abuse

▼ 2.1.3.1 ICMP Abuse

▼ 2.1.3.1.1 ICMP Address
Mask Request/Reply

2.1.3.1.1 ICMP Address Mask Request/Reply

Detection

An ICMP Address Mask Request looks as follows on the wire.

Internet Control Message Protocol

Type: 17 (Address mask request)

Code: 0

Checksum: 0x17a8 [correct]

Identifier (BE): 54615 (0xd557)

Identifier (LE): 22485 (0x57d5)

Sequence number (BE): 512 (0x0200)

Sequence number (LE): 2 (0x0002)

Address mask: 0.0.0.0 (0x00000000)

0000 08 00 27 16 0f 55 08 00 27 0a c9 f5 08 00 45 00 ..'..U.. '.....E.

0010 00 20 3f 0b 00 00 40 01 23 c0 0a 00 02 0f 0a 00 .?...@. #.....

0020 02 04 11 00 17 a8 d5 57 02 00 00 00 00 00 00W

OUTLINE

Timestamps Option

2.1.1.7.1 TCP
Timestamps Option

2.1.1.7.1 TCP
Timestamps Option

2.1.1.7.2 Leveraging
TCP Option Supp...

2.1.1.7.2 Leveraging
TCP Option Supp...

▼ 2.1.2 UDP

2.1.2 UDP

▼ 2.1.3 ICMP

2.1.3.1 ICMP Abuse

▼ 2.1.3.1 ICMP Abuse

▼ 2.1.3.1.1 ICMP Address Mask Request/Reply

2.1.3.1.1 ICMP
Address Mask Re...

2.1.3.1.2 ICMP Timestamp Request/Reply

ICMP can also be used to obtain timestamps from remote systems. We previously covered how valuable a timestamp can be. That being said, as the years progressed many Operating Systems have implemented less accurate timestamps, making this approach unreliable.

An attacker can still use this approach though, as an alternative to ping when things are filtered.

OUTLINE

Timestamps Option

2.1.1.7.1 TCP
Timestamps Option

2.1.1.7.2 Leveraging
TCP Option Supp...

2.1.1.7.2 Leveraging
TCP Option Supp...

▼ 2.1.2 UDP

2.1.2 UDP

▼ 2.1.3 ICMP

2.1.3.1 ICMP Abuse

▼ 2.1.3.1 ICMP Abuse

▼ 2.1.3.1.1 ICMP Address
Mask Request/Reply

2.1.3.1.1 ICMP
Address Mask Re...

▼ 2.1.3.1.2 ICMP
Timestamp Request/R...

2.1.3.1.2 ICMP Timestamp Request/Reply

Detection

An ICMP Timestamp Request looks as follows on the wire.

```
Internet Control Message Protocol
Type: 13 (Timestamp request)
Code: 0
Checksum: 0x7b83 [correct]
Identifier (BE): 15455 (0x3c5f)
Identifier (LE): 24380 (0x5f3c)
Sequence number (BE): 0 (0x0000)
Sequence number (LE): 0 (0x0000)
Originate timestamp: 18 hours, 46 minutes, 21.718 seconds after midnight UTC
Receive timestamp: 0 seconds after midnight UTC
Transmit timestamp: 0 seconds after midnight UTC
```

OUTLINE

Timestamps Option...

2.1.1.7.2 Leveraging
TCP Option Supp...

2.1.1.7.2 Leveraging
TCP Option Supp...

▼ 2.1.2 UDP

2.1.2 UDP

▼ 2.1.3 ICMP

2.1.3.1 ICMP Abuse

▼ 2.1.3.1 ICMP Abuse

▼ 2.1.3.1.1 ICMP Address Mask Request/Reply

2.1.3.1.1 ICMP
Address Mask Re...

▼ 2.1.3.1.2 ICMP Timestamp Request/R...

2.1.3.1.2 ICMP
Timestamp Requ...

2.1.3.1.2 ICMP Timestamp Request/Reply

Detection

An ICMP Timestamp Reply looks as follows on the wire.

```
Internet Control Message Protocol
Type: 14 (Timestamp reply)
Code: 0
Checksum: 0xce3e [correct]
Identifier (BE): 15455 (0x3c5f)
Identifier (LE): 24380 (0x5f3c)
Sequence number (BE): 0 (0x0000)
Sequence number (LE): 0 (0x0000)
Originate timestamp: 18 hours, 46 minutes, 21.718 seconds after midnight UTC
Receive timestamp: 13 hours, 19 minutes, 53.670 seconds after midnight UTC
Transmit timestamp: 13 hours, 19 minutes, 53.670 seconds after midnight UTC
```

OUTLINE

2.1.1.7.2 Leveraging
TCP Option Supp...

▼ 2.1.2 UDP

2.1.2 UDP

▼ 2.1.3 ICMP

2.1.3.1 ICMP Abuse

▼ 2.1.3.1 ICMP Abuse

▼ 2.1.3.1.1 ICMP Address
Mask Request/Reply

2.1.3.1.1 ICMP
Address Mask Re...

▼ 2.1.3.1.2 ICMP
Timestamp Request/R...

2.1.3.1.2 ICMP
Timestamp Requ...

2.1.3.1.2 ICMP
Timestamp Requ...

2.1.3.1 ICMP Timestamp Request/Reply

Detection

A great paper showcasing ICMP usage during scanning activities can be found below.

https://www.cs.dartmouth.edu/~sergey/me/netreads/ICMP_Scanning_v3.0.pdf

OUTLINE

▼ 2.1.2 UDP

2.1.2 UDP

▼ 2.1.3 ICMP

2.1.3.1 ICMP Abuse

▼ 2.1.3.1 ICMP Abuse

▼ 2.1.3.1.1 ICMP Address Mask Request/Reply

2.1.3.1.1 ICMP Address Mask Re...

▼ 2.1.3.1.2 ICMP Timestamp Request/R...

2.1.3.1.2 ICMP Timestamp Requ...

2.1.3.1.2 ICMP Timestamp Requ...

▼ 2.1.3.1 ICMP Timestamp Request/Reply

2.1.3.1.3 Smurf Attack

ICMP has been used in the past to execute devastating Distributed Denial of Service (DDoS) attacks. Such an attack, was the notorious Smurf attack.

OUTLINE

2.1.2 UDP

▼ 2.1.3 ICMP

2.1.3.1 ICMP Abuse

▼ 2.1.3.1 ICMP Abuse

▼ 2.1.3.1.1 ICMP Address Mask Request/Reply

2.1.3.1.1 ICMP Address Mask Re...

▼ 2.1.3.1.2 ICMP Timestamp Request/R...

2.1.3.1.2 ICMP Timestamp Requ...

2.1.3.1.2 ICMP Timestamp Requ...

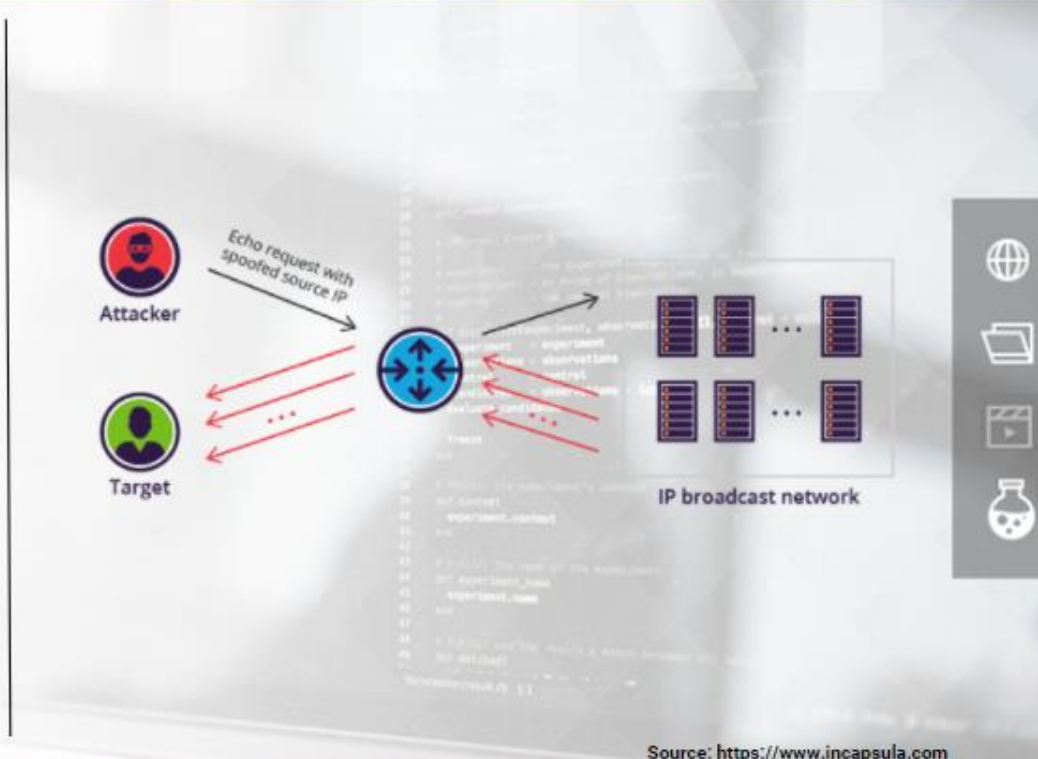
▼ 2.1.3.1 ICMP Timestamp Request/Reply

▼ 2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

A Smurf attack is executed as follows:

1. A fake Echo request is generated containing a spoofed source IP, which is actually the target server address
2. The request is sent to an intermediate IP broadcast network (The intermediate site/router has to allow ICMP echo requests to broadcast address)
3. The request is transmitted to all of the network hosts on the network
4. All hosts on the network that respond, will send an ICMP echo reply to the target server
5. If the number of ICMP responses is big enough, denial of service or performance degradation is caused to the target server.



Source: <https://www.incapsula.com>

OUTLINE

▼ 2.1.3 ICMP

2.1.3.1 ICMP Abuse

▼ 2.1.3.1 ICMP Abuse

▼ 2.1.3.1.1 ICMP Address Mask Request/Reply

2.1.3.1.1 ICMP Address Mask Re...

▼ 2.1.3.1.2 ICMP Timestamp Request/R...

2.1.3.1.2 ICMP Timestamp Requ...

2.1.3.1.2 ICMP Timestamp Requ...

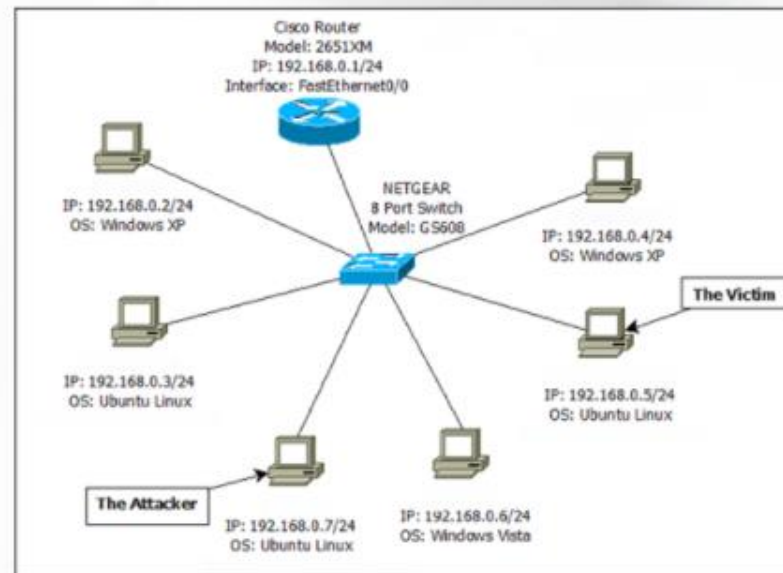
▼ 2.1.3.1 ICMP Timestamp Request/Reply

▼ 2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

Suppose, an environment is set up as follows and the attacker (192.168.0.7) performs a Smurf attack against the victim (192.168.0.5)



Source: <http://www.uwindsor.ca/>

OUTLINE

2.1.3.1 ICMP Abuse

▼ 2.1.3.1 ICMP Abuse

▼ 2.1.3.1.1 ICMP Address Mask Request/Reply

2.1.3.1.1 ICMP Address Mask Re...

▼ 2.1.3.1.2 ICMP Timestamp Request/R...

2.1.3.1.2 ICMP Timestamp Requ...

2.1.3.1.2 ICMP Timestamp Requ...

▼ 2.1.3.1 ICMP Timestamp Request/Reply

▼ 2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

Detection

The Smurf attack on the wire looks as follows.

16078	10938.022614	192.168.0.5	192.168.0.255	ICMP	Echo (ping) request
16079	10938.024323	CadmusCo_89:53:1b	Broadcast	ARP	who has 192.168.0.5? Tell 192.168.0.3
16080	10938.024735	CadmusCo_79:4c:f0	CadmusCo_89:53:1b	ARP	192.168.0.5 is at 08:00:27:29:4c:f0
16081	10938.025020	192.168.0.3	192.168.0.5	ICMP	Echo (ping) reply

7576	5964.414514	192.168.0.7	192.168.0.5	ICMP	echo (ping) request
7577	5964.415480	192.168.0.5	192.168.0.7	ICMP	echo (ping) reply
7578	5964.427527	192.168.0.5	192.168.0.255	ICMP	echo (ping) request
7579	5964.438459	CadmusCo_89:53:1b	Broadcast	ARP	who has 192.168.0.5? Tell 192.168.0.3
7580	5964.438896	CadmusCo_79:4c:f0	CadmusCo_89:53:1b	ARP	192.168.0.5 is at 08:00:27:29:4c:f0
7581	5964.439175	192.168.0.1	192.168.0.5	ICMP	echo (ping) reply
7582	5964.439269	192.168.0.3	192.168.0.5	ICMP	echo (ping) reply
7583	5964.728886	192.168.0.6	192.168.0.255	NBNS	Name query NB <00>
7584	5964.925831	192.168.0.6	192.168.0.255	NBNS	Name query NB <20>
7585	5965.415810	192.168.0.7	192.168.0.5	ICMP	echo (ping) request
7586	5965.777136	F680:e07e:58fd:8660:ff02::1:3	LLMNR	Standard query A	
7587	5965.777148	192.168.0.6	224.0.0.252	LLMNR	Standard query A
7588	5965.977505	192.168.0.6	192.168.0.255	NBNS	Name query NB <00>
7589	5966.242515	0m_of:2a:da	Spanning-tree-(for-br	STP	Conf. Root = 32768/0/02:08:29:df:2a:da Cost = 0 Port = 0x8000
7590	5966.426337	192.168.0.7	192.168.0.5	ICMP	echo (ping) request
7591	5966.727822	192.168.0.6	192.168.0.255	NBNS	Name query NB <00>
7592	5967.432405	192.168.0.7	192.168.0.5	ICMP	echo (ping) request

Source: <http://www.uwindsor.ca/>

OUTLINE

- ▼ 2.1.3.1 ICMP Abuse
 - ▼ 2.1.3.1.1 ICMP Address Mask Request/Reply
 - 2.1.3.1.1 ICMP Address Mask Re...
 - ▼ 2.1.3.1.2 ICMP Timestamp Request/R...
 - 2.1.3.1.2 ICMP Timestamp Requ...
 - 2.1.3.1.2 ICMP Timestamp Requ...
 - ▼ 2.1.3.1 ICMP Timestamp Request/Reply
 - ▼ 2.1.3.1.3 Smurf Attack
 - 2.1.3.1.3 Smurf Attack
 - 2.1.3.1.3 Smurf Attack
 - 2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

Detection

The Smurf attack on the wire looks as follows.

The spoofed packet was injected and broadcasted successfully

16078	10938.022614	192.168.0.5	192.168.0.255	ICMP	Echo (ping) request
16079	10938.024323	CadmusCo_89:53:1b	Broadcast	ARP	who has 192.168.0.5? Tell 192.168.0.5
16080	10938.024735	CadmusCo_79:4c:f0	CadmusCo_89:53:1b	ARP	192.168.0.5 is at 08:00:27:29:4c:f0
16081	10938.025020	192.168.0.3	192.168.0.5	ICMP	Echo (ping) reply

The victim no longer replies to Echo Requests

7576	5964.414514	192.168.0.7	192.168.0.5	ICMP	echo (ping) request
7577	5964.415480	192.168.0.5	192.168.0.7	ICMP	echo (ping) reply
7578	5964.427527	192.168.0.5	192.168.0.255	ICMP	echo (ping) request
7579	5964.438459	CadmusCo_89:53:1b	Broadcast	ARP	who has 192.168.0.5? Tell 192.168.0.3
7580	5964.438896	CadmusCo_29:4c:f0	CadmusCo_89:53:1b	ARP	192.168.0.5 is at 08:00:27:29:4c:f0
7581	5964.439175	192.168.0.1	192.168.0.5	ICMP	echo (ping) reply
7582	5964.439269	192.168.0.3	192.168.0.5	ICMP	echo (ping) reply
7583	5964.728886	192.168.0.6	192.168.0.255	NBNS	Name query NB <00>
7584	5964.925831	192.168.0.6	192.168.0.255	NBNS	Name query NB <20>
7585	5965.415810	192.168.0.7	192.168.0.5	ICMP	echo (ping) request
7586	5965.777136	F680::e07e:58fd:8660	FF02::1:3	LLMNR	Standard query A
7587	5965.777148	192.168.0.6	224.0.0.252	LLMNR	Standard query A
7588	5965.977505	192.168.0.6	192.168.0.255	NBNS	Name query NB <00>
7589	5966.245515	0m_of:2a:da	Spanning-tree-(for-br	STP	Conf. Root = 32768/0/02:08:29:df:2a:da Cost = 0 Port = 0x8000
7590	5966.426337	192.168.0.7	192.168.0.5	ICMP	echo (ping) request
7591	5966.727822	192.168.0.6	192.168.0.255	NBNS	Name query NB <00>
7592	5967.432405	192.168.0.7	192.168.0.5	ICMP	echo (ping) request

Source: <http://www.uwindsor.ca/>

OUTLINE

2.1.3.1.1 ICMP Address Mask Request/Reply

2.1.3.1.1 ICMP Address Mask Re...

2.1.3.1.2 ICMP Timestamp Request/R...

2.1.3.1.2 ICMP Timestamp Requ...

2.1.3.1.2 ICMP Timestamp Requ...

2.1.3.1 ICMP Timestamp Request/Reply

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

Detection

The IPv6 variant of the Smurf attack on the wire looks as follows.

245409	163.369213000	2001:abc:1:1:a00:27ff:fed0:c21a	ff02::1	ICMPv6	78 Echo (ping) request id=0xface, seq=47806
245410	163.369225000	2001:abc:1:1:dd90:9266:fcdf:d760	2001:abc:1:1:a00:27ff:fed0:c21a	ICMPv6	78 Echo (ping) reply id=0xface, seq=47806
245411	163.369267000	2001:abc:1:1:a00:27ff:fed0:c21a	ff02::1	ICMPv6	78 Echo (ping) request id=0xface, seq=47806
245412	163.369270000	2001:abc:1:1:dd90:9266:fcdf:d760	2001:abc:1:1:a00:27ff:fed0:c21a	ICMPv6	78 Echo (ping) reply id=0xface, seq=47806
245413	163.369679000	2001:abc:1:1:a00:27ff:fed0:c21a	ff02::1	ICMPv6	78 Echo (ping) request id=0xface, seq=47806
245414	163.369695000	2001:abc:1:1:dd90:9266:fcdf:d760	2001:abc:1:1:a00:27ff:fed0:c21a	ICMPv6	78 Echo (ping) reply id=0xface, seq=47806
245415	163.369952000	2001:abc:1:1:a00:27ff:fed0:c21a	ff02::1	ICMPv6	78 Echo (ping) request id=0xface, seq=47806
245416	163.369966000	2001:abc:1:1:dd90:9266:fcdf:d760	2001:abc:1:1:a00:27ff:fed0:c21a	ICMPv6	78 Echo (ping) reply id=0xface, seq=47806

OUTLINE

mask request reply

2.1.3.1.1 ICMP
Address Mask Re...

2.1.3.1.2 ICMP
Timestamp Request/R...

2.1.3.1.2 ICMP
Timestamp Requ...

2.1.3.1.2 ICMP
Timestamp Requ...

2.1.3.1 ICMP Timestamp
Request/Reply

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf
Attack

2.1.3.1.3 Smurf
Attack

2.1.3.1.3 Smurf
Attack

2.1.3.1.3 Smurf
Attack

2.1.3.1.3 Smurf
Attack

2.1.3.1.3 Smurf Attack

Detection

The IPv6 variant of the Smurf attack on the wire looks as follows.

The spoofed packet was injected and broadcasted successfully

245409	163.369213000	2001:abc:1:1:a00:27ff:fed0:c21a	ff02::1	ICMPv6	78 Echo (ping) request id=0xface, seq=47806
245410	163.369225000	2001:abc:1:1:dd90:9266:fcdf:d760	2001:abc:1:1:a00:27ff:fed0:c21a	ICMPv6	78 Echo (ping) reply id=0xface, seq=47806
245411	163.369267000	2001:abc:1:1:a00:27ff:fed0:c21a	ff02::1	ICMPv6	78 Echo (ping) request id=0xface, seq=47806
245412	163.369270000	2001:abc:1:1:dd90:9266:fcdf:d760	2001:abc:1:1:a00:27ff:fed0:c21a	ICMPv6	78 Echo (ping) reply id=0xface, seq=47806
245413	163.369679000	2001:abc:1:1:a00:27ff:fed0:c21a	ff02::1	ICMPv6	78 Echo (ping) request id=0xface, seq=47806
245414	163.369695000	2001:abc:1:1:dd90:9266:fcdf:d760	2001:abc:1:1:a00:27ff:fed0:c21a	ICMPv6	78 Echo (ping) reply id=0xface, seq=47806
245415	163.369952000	2001:abc:1:1:a00:27ff:fed0:c21a	ff02::1	ICMPv6	78 Echo (ping) request id=0xface, seq=47806
245416	163.369966000	2001:abc:1:1:dd90:9266:fcdf:d760	2001:abc:1:1:a00:27ff:fed0:c21a	ICMPv6	78 Echo (ping) reply id=0xface, seq=47806

OUTLINE

2.1.3.1.2 ICMP Timestamp Request/R...

2.1.3.1.2 ICMP Timestamp Requ...

2.1.3.1.2 ICMP Timestamp Requ...

2.1.3.1 ICMP Timestamp Request/Reply

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.4 ICMP Tunneling

It should be noted, that ICMP can be misused to create a covert channel of communication. This can be achieved through ill-intended ICMP tunneling.

Numerous ICMP tunneling solutions exists, but attackers seem to prefer the [ptunnel](http://freshmeat.sourceforge.net/projects/ptunnel/) one.

OUTLINE

- 2.1.3.1.2 ICMP Timestamp Request/Reply
- 2.1.3.1.2 ICMP Timestamp Request/Reply
- ▼ 2.1.3.1 ICMP Timestamp Request/Reply
 - ▼ 2.1.3.1.3 Smurf Attack
 - 2.1.3.1.3 Smurf Attack
 - 2.1.3.1.3 Smurf Attack
 - 2.1.3.1.3 Smurf Attack
 - 2.1.3.1.3 Smurf Attack
 - 2.1.3.1.3 Smurf Attack
 - 2.1.3.1.3 Smurf Attack
 - ▼ 2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

Consider the *ptunnel.pcap* file found in this module's resources. Try to identify any abnormalities.

OUTLINE

2.1.3.1 ICMP Timestamp Request/Reply

2.1.3.1.2 ICMP Timestamp Request/Reply

▼ 2.1.3.1 ICMP Timestamp Request/Reply

▼ 2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

▼ 2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

Detection

1	0.000000	192.168.1.100	192.168.1.4	ICMP	70 Echo (ping) request	id=0x9a32, seq=0/0, ttl=64 (reply in 2)
2	0.000046	192.168.1.4	192.168.1.100	ICMP	70 Echo (ping) reply	id=0x9a32, seq=0/0, ttl=64 (request in 1)
3	0.014989	192.168.1.4	192.168.1.100	ICMP	82 Echo (ping) reply	id=0x9a32, seq=0/0, ttl=64
4	0.015246	192.168.1.100	192.168.1.4	ICMP	82 Echo (ping) request	id=0x9a32, seq=1/256, ttl=64 (reply in 5)
5	0.015262	192.168.1.4	192.168.1.100	ICMP	82 Echo (ping) reply	id=0x9a32, seq=1/256, ttl=64 (request in 4)
6	0.015319	192.168.1.4	192.168.1.100	ICMP	70 Echo (ping) reply	id=0x9a32, seq=1/256, ttl=64
7	0.015439	192.168.1.4	192.168.1.100	ICMP	94 Echo (ping) reply	id=0x9a32, seq=2/512, ttl=64
8	0.015612	192.168.1.100	192.168.1.4	ICMP	170 Echo (ping) request	id=0x9a32, seq=2/512, ttl=64 (reply in 9)
9	0.015622	192.168.1.4	192.168.1.100	ICMP	170 Echo (ping) reply	id=0x9a32, seq=2/512, ttl=64 (request in 8)
10	0.017822	192.168.1.4	192.168.1.100	ICMP	86 Echo (ping) reply	id=0x9a32, seq=3/768, ttl=64
11	0.017958	192.168.1.100	192.168.1.4	ICMP	94 Echo (ping) request	id=0x9a32, seq=3/768, ttl=64 (reply in 12)
12	0.017977	192.168.1.4	192.168.1.100	ICMP	94 Echo (ping) reply	id=0x9a32, seq=3/768, ttl=64 (request in 11)
13	0.018171	192.168.1.4	192.168.1.100	ICMP	74 Echo (ping) reply	id=0x9a32, seq=4/1024, ttl=64
14	0.018282	192.168.1.100	192.168.1.4	ICMP	74 Echo (ping) request	id=0x9a32, seq=4/1024, ttl=64 (reply in 15)
15	0.018295	192.168.1.4	192.168.1.100	ICMP	74 Echo (ping) reply	id=0x9a32, seq=4/1024, ttl=64 (request in 14)
16	0.018440	192.168.1.4	192.168.1.100	ICMP	690 Echo (ping) reply	id=0x9a32, seq=5/1280, ttl=64



OUTLINE

icmp_timestamp_request

2.1.3.1 ICMP Timestamp Request/Reply

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

Detection

1	0.000000	192.168.1.100	192.168.1.4	ICMP	70 Echo (ping) request	id=0x9a32, seq=0/0, ttl=64 (reply in 2)
2	0.000046	192.168.1.4	192.168.1.100	ICMP	70 Echo (ping) reply	id=0x9a32, seq=0/0, ttl=64 (request in 1)
3	0.014989	192.168.1.4	192.168.1.100	ICMP	82 Echo (ping) reply	id=0x9a32, seq=0/0, ttl=64
4	0.015246	192.168.1.100	192.168.1.4	ICMP	82 Echo (ping) request	id=0x9a32, seq=1/256, ttl=64 (reply in 5)
5	0.015262	192.168.1.4	192.168.1.100	ICMP	82 Echo (ping) reply	id=0x9a32, seq=1/256, ttl=64 (request in 4)
6	0.015319	192.168.1.4	192.168.1.100	ICMP	70 Echo (ping) reply	id=0x9a32, seq=1/256, ttl=64
7	0.015439	192.168.1.4	192.168.1.100	ICMP	94 Echo (ping) reply	id=0x9a32, seq=2/512, ttl=64
8	0.015612	192.168.1.100	192.168.1.4	ICMP	170 Echo (ping) request	id=0x9a32, seq=2/512, ttl=64 (reply in 9)
9	0.015622	192.168.1.4	192.168.1.100	ICMP	170 Echo (ping) reply	id=0x9a32, seq=2/512, ttl=64 (request in 8)
10	0.017822	192.168.1.4	192.168.1.100	ICMP	86 Echo (ping) reply	id=0x9a32, seq=3/768, ttl=64
11	0.017958	192.168.1.100	192.168.1.4	ICMP	94 Echo (ping) request	id=0x9a32, seq=3/768, ttl=64 (reply in 12)
12	0.017977	192.168.1.4	192.168.1.100	ICMP	94 Echo (ping) reply	id=0x9a32, seq=3/768, ttl=64 (request in 11)
13	0.018171	192.168.1.4	192.168.1.100	ICMP	74 Echo (ping) reply	id=0x9a32, seq=4/1024, ttl=64
14	0.018282	192.168.1.100	192.168.1.4	ICMP	74 Echo (ping) request	id=0x9a32, seq=4/1024, ttl=64 (reply in 15)
15	0.018295	192.168.1.4	192.168.1.100	ICMP	74 Echo (ping) reply	id=0x9a32, seq=4/1024, ttl=64 (request in 14)
16	0.018440	192.168.1.4	192.168.1.100	ICMP	690 Echo (ping) reply	id=0x9a32, seq=5/1280, ttl=64

- The capture file seems imbalanced as far as the amount of Echo requests and Echo replies is concerned. You can see two or more replies for each request
- An Echo request and the associated Echo reply should have the same length (have identical payloads). This isn't the case in this capture file.
- The payload size in some ICMP packets is a lot bigger than normal. For example, see packet #16.

OUTLINE

request reply

▼ 2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

▼ 2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

Detection

Let's dig deeper...

First, download [ptunnel](#) and then, open the *ptunnel.h* file using a text editor. Around line #191, you will identify that the ptunnel header begins with a magic number, a short unique sequence of bytes that lets the recipient know that the ICMP payload should be treated as ptunnel traffic. This magic number is 0xD5200880, by default.

```
191 #define PTUNNEL_MAGIC 0xD5200880
192 // Resend packets after this interval (in seconds)
193 #define KRESEND_INTERVAL 1.5
194
195 /* ping_tunnel_pkt_t: This data structure represents the header of a ptunnel
196    packet, consisting of a magic number, the tunnel's destination IP and port,
197    as well as some other fields. Note that the dest IP and port is only valid
198    in packets from the client to the proxy.
199 */
200 typedef struct {
201     uint32_t magic; // magic number, used to identify ptunnel packets.
202     struct { // destination IP and port (used by proxy to figure
203         uint32_t ip; // out where to tunnel to)
204         uint16_t port; // current connection state: see constants above.
205     } dst; // sequence number of last packet received from other end
206     uint32_t seq_no; // length of data buffer
207     uint16_t data_len; // sequence number of this packet
208     uint32_t id_no; // id number, used to separate different tunnels from each other
209     char data[0]; // optional data buffer
210 } __attribute__((packed)) ping_tunnel_pkt_t;
```

OUTLINE

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

▼ 2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

Detection

It looks like all ICMP packets of this capture file include the previously mentioned ptunnel magic value (0xD5200880), inside the ICMP payload. The yellow rectangles designate the ICMP header session identifier.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.100	192.168.1.4	ICMP	76	Echo (ping) request id=0x9a32, seq=0/0, ttl=64 (reply in 2)
2	0.000040	192.168.1.4	192.168.1.100	ICMP	76	Echo (ping) reply id=0x9a32, seq=0/0, ttl=64 (request in 1)
3	0.014960	192.168.1.4	192.168.1.100	ICMP	82	Echo (ping) reply id=0x9a32, seq=0/0, ttl=64
4	0.035200	192.168.1.100	192.168.1.4	ICMP	82	Echo (ping) request id=0x9a32, seq=1/256, ttl=64 (reply in 3)
5	0.035260	192.168.1.4	192.168.1.100	ICMP	82	Echo (ping) reply id=0x9a32, seq=1/256, ttl=64 (request in 4)
6	0.035310	192.168.1.4	192.168.1.100	ICMP	76	Echo (ping) reply id=0x9a32, seq=1/256, ttl=64

Code	Checksum	Identifier (BE)	Identifier (LE)	Sequence number (BE)	Sequence number (LE)	Response frame
0	8a5d3 [correct]	39474 (0x9a32)	4904 (0x129a)	0 (0x0000)	0 (0x0000)	2

Offset	0000	0004	0008	000c	0010	0014	0018	001c	0020	0024	0028	002c	0030	0034	0038	003c	0040
0000	00	50	56	b1	3a	08	00	50	56	b1	33	b1	00	70	45	80	..PV...P.V.3...E.
0004	00	30	90	52	40	00	40	01	20	09	00	10	01	01	c0	a8	..0.50.0.d..
0008	01	04	00	00	e3	03	0a	32	00	00	00	00	00	00	00	00	2.....
000c	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

OUTLINE

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

▼ 2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

Detection

In addition, there are indications that a whole session, between an attacker and a compromised machine, is being concealed through ICMP traffic (ICMP tunneling).

[illegible]

OUTLINE

FIELDWORK

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

▼ 2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

Detection

The fact that a whole session, between an attacker and a compromised machine, is being concealed through ICMP traffic (ICMP tunneling) can be also identified by using the *strings* Linux command against the capture file as follows.

```
>> strings ptunnel.pcap
```

```
root@kali:~/Desktop/IHRP_PCAPs/Part 2# strings ptunnel.pcap
0060
8080
P090
38400,38400
nku-kali.hh.nku.edu:1
DISPLAY
nku-kali.hh.nku.edu:1
xterm-256color
38400,38400
nku-kali.hh.nku.edu:1
DISPLAY
nku-kali.hh.nku.edu:1
xterm-256color
H0;@
<0=0
2
metasploitable2
Warning: Never expose this VM to an untrusted network!
Contact: msfdev[at]metasploit.com
Login with msfadmin/msfadmin to get started
metasploitable login: 7L
:13@
:1E@
:1W@
81e@
:1l@
:1w@
:1y@
2Password: 9L
```

OUTLINE

FILEDOW

2.1.3.1.3 Smurf
Attack

2.1.3.1.3 Smurf
Attack

2.1.3.1.3 Smurf
Attack

▼ 2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP
Tunneling

2.1.3.1.4 ICMP
Tunneling

2.1.3.1.4 ICMP
Tunneling

2.1.3.1.4 ICMP
Tunneling

2.1.3.1.4 ICMP
Tunneling

2.1.3.1.4 ICMP
Tunneling

2.1.3.1.4 ICMP
Tunneling

2.1.3.1.5 Abusing ICMP Redirect

When two machines want to communicate with each other the router has to find the shortest path between them.

If there is an alternate path between the two machines and if it is shorter than the previous path, then, the router will send an ICMP redirect packet to the sender machine.

OUTLINE

FILED

2.1.3.1.3 Smurf Attack

2.1.3.1.3 Smurf Attack

▼ 2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

▼ 2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

This ICMP redirect packet will instruct the sender to change its routing table so that it uses the shortest path.

However, such ICMP redirect packets can be forged by an attacker and make the sender host redirect its packet to an attacker-controlled or non-existing destination.

OUTLINE

2.1.3.1.1

2.1.3.1.3 Smurf Attack

▼ 2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

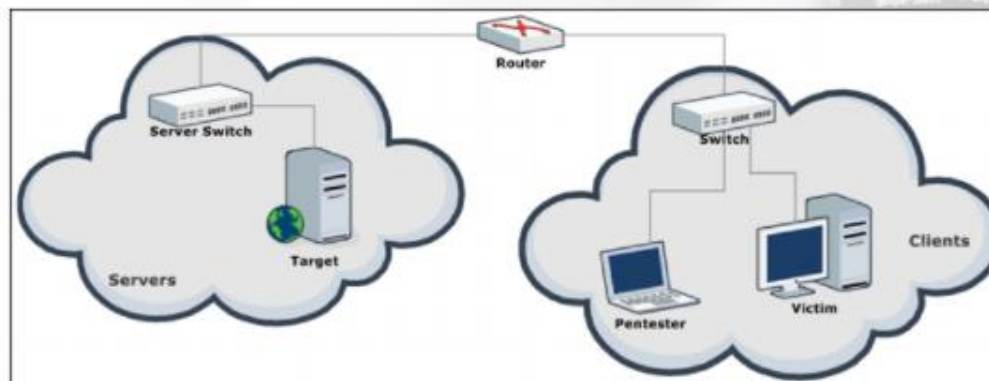
2.1.3.1.4 ICMP Tunneling

▼ 2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

Suppose, an environment is set up as follows. The network has been hardened to prevent successful execution of ARP- spoofing-based attacks. Somehow, the pentester residing in the client network, managed to sniff client traffic and capture clear-text credentials.



OUTLINE

▼ 2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

▼ 2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

Consider the *client_network.pcap* file found in this module's resources. Try to identify how the pentester managed to sniff client traffic, in spite of the anti-ARP-poisoning measures.

Note: The majority of TCP traffic has been removed.

OUTLINE

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

▼ 2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

Detection

1	0.000000	72:9b:2f:a0:90:91	Broadcast	ARP	60	Who has 10.100.13.126? Tell 10.100.13.20
2	0.000022	Vmware_a1:cb:34	72:9b:2f:a0:90:91	ARP	42	10.100.13.126 is at 00:50:56:a1:cb:34
3	0.251032	10.100.13.1	10.100.13.126	ICMP	82	Redirect (Redirect for host)
4	0.333845	10.100.13.1	10.100.13.126	ICMP	82	Redirect (Redirect for host)
5	0.397539	10.100.13.1	10.100.13.126	ICMP	82	Redirect (Redirect for host)
6	0.468985	10.100.13.1	10.100.13.126	ICMP	82	Redirect (Redirect for host)
7	0.555181	10.100.13.1	10.100.13.126	ICMP	82	Redirect (Redirect for host)
8	0.637849	10.100.13.1	10.100.13.126	ICMP	82	Redirect (Redirect for host)
9	0.722346	10.100.13.1	10.100.13.126	ICMP	82	Redirect (Redirect for host)
10	0.811009	10.100.13.1	10.100.13.126	ICMP	82	Redirect (Redirect for host)
11	0.900367	10.100.13.1	10.100.13.126	ICMP	82	Redirect (Redirect for host)
12	0.968061	10.100.13.1	10.100.13.126	ICMP	82	Redirect (Redirect for host)

▶ Frame 3: 82 bytes on wire (656 bits), 82 bytes captured (656 bits)
▶ Ethernet II, Src: 72:9b:2f:a0:90:91 (72:9b:2f:a0:90:91), Dst: Vmware_a1:cb:34 (00:50:56:a1:cb:34)
▶ Internet Protocol Version 4, Src: 10.100.13.1, Dst: 10.100.13.126
▼ Internet Control Message Protocol
Type: 5 (Redirect)
Code: 1 (Redirect for host)
Checksum: 0x3dfe [correct]
[Checksum Status: Good]
Gateway address: 10.100.13.20
▶ Internet Protocol Version 4, Src: 10.100.13.126, Dst: 10.23.56.100
▶ Transmission Control Protocol, Src Port: 55555, Dst Port: 80

OUTLINE

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

Detection

1 0.000000	72:9b:2f:a0:90:91	Broadcast	ARP	60 Who has 10.100.13.126? Tell 10.100.13.20
2 0.000022	Vmware_a1:cb:34	72:9b:2f:a0:90:91	ARP	42 10.100.13.126 is at 00:50:56:a1:cb:34
3 0.251032	10.100.13.1	10.100.13.126	ICMP	82 Redirect (Redirect for host)
4 0.333845	10.100.13.1	10.100.13.126	ICMP	82 Redirect (Redirect for host)
5 0.397539	10.100.13.1	10.100.13.126	ICMP	82 Redirect (Redirect for host)
6 0.468985	10.100.13.1	10.100.13.126	ICMP	82 Redirect (Redirect for host)
7 0.555181	10.100.13.1	10.100.13.126	ICMP	82 Redirect (Redirect for host)
8 0.637849	10.100.13.1	10.100.13.126	ICMP	82 Redirect (Redirect for host)
9 0.722346	10.100.13.1	10.100.13.126	ICMP	82 Redirect (Redirect for host)
10 0.811009	10.100.13.1	10.100.13.126	ICMP	82 Redirect (Redirect for host)
11 0.900367	10.100.13.1	10.100.13.126	ICMP	82 Redirect (Redirect for host)
12 0.968061	10.100.13.1	10.100.13.126	ICMP	82 Redirect (Redirect for host)

▶ Frame 3: 82 bytes on wire (656 bits), 82 bytes captured (656 bits)
▶ Ethernet II, Src: 72:9b:2f:a0:90:91 (72:9b:2f:a0:90:91), Dst: Vmware_a1:cb:34 (00:50:56:a1:cb:34)
▶ Internet Protocol Version 4, Src: 10.100.13.1, Dst: 10.100.13.126
▼ Internet Control Message Protocol
Type: 5 (Redirect)
Code: 1 (Redirect for host)
Checksum: 0x3dfe [correct]
[Checksum Status: Good]
Gateway address: 10.100.13.20
▶ Internet Protocol Version 4, Src: 10.100.13.126, Dst: 10.23.56.100
▶ Transmission Control Protocol, Src Port: 55555, Dst Port: 80

- A large number of ICMP Redirect packets exist. In those packets the router instructs the client 10.100.13.126 to make a change in its routing table to use the gateway 10.100.13.20 for all subsequent packets. At this moment, you should check if the gateway 10.100.13.20 is a legitimate gateway.
- If you filter the whole capture file, based on the MAC address of the router (10.100.13.1) [eth.src==72:9b:2f:a0:90:91], you will notice that this MAC address is associated with the 10.100.13.20 machine.
- Even though it is not clearly visible in the capture file, every HTTP request can now be sniffed by the 10.100.13.20 host, as a result of the ICMP Redirect attack.
- So, it looks like the penetration tester crafted malicious ICMP Redirect packets, spoofing the router, and mounted an ICMP Redirect attack.

OUTLINE

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

▼ 2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1 Analyzing & Detecting Transport Layer Attacks

This is the end of the *Analyzing & Detecting Transport Layer Attacks* part.

Moving forward, we will show you how to analyze common application protocol traffic and attacks.

```
15 # Return empty list
16
17 # Return empty list
18
19 # Return empty list
20
21 # Return empty list
22
23 # Return empty list
24
25 # Return empty list
26
27 # Return empty list
28
29 # Return empty list
30
31 # Return empty list
32
33 # Return empty list
34
35 # Return empty list
36
37 # Return empty list
38
39 # Return empty list
40
41 # Return empty list
42
43 # Return empty list
44
45 # Return empty list
46
47 # Return empty list
48
49 # Return empty list
50
51 # Return empty list
52
53 # Return empty list
54
55 # Return empty list
56
57 # Return empty list
58
59 # Return empty list
60
61 # Return empty list
62
63 # Return empty list
64
65 # Return empty list
66
67 # Return empty list
68
69 # Return empty list
70
71 # Return empty list
72
73 # Return empty list
74
75 # Return empty list
76
77 # Return empty list
78
79 # Return empty list
80
81 # Return empty list
82
83 # Return empty list
84
85 # Return empty list
86
87 # Return empty list
88
89 # Return empty list
90
91 # Return empty list
92
93 # Return empty list
94
95 # Return empty list
96
97 # Return empty list
98
99 # Return empty list
100
```



OUTLINE

2.1 Analyzing & Detecting

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1 Analyzing & Detecting
Transport Layer Attacks

IHRP

2.2

Analyzing Common Application Protocol Traffic & Attacks



OUTLINE

2.1 Analyzing & Detecting

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

▼ 2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1 Analyzing & Detecting Transport Layer Attacks

▼ 2.2 Analyzing Common Application Protocol Traffic & Attacks

2.2 Analyzing Common Application Protocol Traffic & Attacks

We eventually reached the Application layer of the TCP/IP model. In this part of the module, we will focus our attention on Microsoft-specific protocols, HTTP(S), SMTP and DNS.

We will analyze them on the wire and also see attacks that either leverage them or misuse them.

```

37  def test_initialize_candidates(self, observations = 10, controls = 10):
38      super_test = super(self, self)
39      observations = observations
40      controls = controls
41      candidates = observations + controls/2
42      self.initialize_candidates
43
44      # test
45
46      # check if the initialize_candidates method
47      self.initialize_candidates
48      self.initialize_candidates
49
50      # check if the initialize_candidates method
51      self.initialize_candidates
52      self.initialize_candidates
53
54      # check if the initialize_candidates method
55      self.initialize_candidates
56      self.initialize_candidates
57
58      # check if the initialize_candidates method
59      self.initialize_candidates
60      self.initialize_candidates
61
62      # check if the initialize_candidates method
63      self.initialize_candidates
64      self.initialize_candidates
65
66      # check if the initialize_candidates method
67      self.initialize_candidates
68      self.initialize_candidates
69
70      # check if the initialize_candidates method
71      self.initialize_candidates
72      self.initialize_candidates
73
74      # check if the initialize_candidates method
75      self.initialize_candidates
76      self.initialize_candidates
77
78      # check if the initialize_candidates method
79      self.initialize_candidates
80      self.initialize_candidates
81
82      # check if the initialize_candidates method
83      self.initialize_candidates
84      self.initialize_candidates
85
86      # check if the initialize_candidates method
87      self.initialize_candidates
88      self.initialize_candidates
89
90      # check if the initialize_candidates method
91      self.initialize_candidates
92      self.initialize_candidates
93
94      # check if the initialize_candidates method
95      self.initialize_candidates
96      self.initialize_candidates
97
98      # check if the initialize_candidates method
99      self.initialize_candidates
100     self.initialize_candidates

```



OUTLINE

2008.05.15

2.1.3.1.4 ICMP Tunneling

2.1.3.1.4 ICMP Tunneling

- 2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1 Analyzing & Detecting Transport Layer Attacks

2.2 Analyzing Common Application Protocol Traffic & Attacks

2.2 Analyzing Common Application Protocol Traffic & At...

2.2.1 Microsoft-specific Protocols

Undoubtedly, Windows is the most widely used Operating System. All those Windows hosts out there, use some universal and Microsoft-specific/Microsoft-implemented protocols. Let's go through them and see how they look in the wire.

OUTLINE

- 2.1 Analyzing & Detecting Transport Layer Attacks
 - 2.1.3.1.4 ICMP Tunneling
 - 2.1.3.1.5 Abusing ICMP Redirect
 - 2.1.3.1.5 Abusing ICMP Redirect
 - 2.1.3.1.5 Abusing ICMP Redirect
 - 2.1.3.1.5 Abusing ICMP Redirect
 - 2.1.3.1.5 Abusing ICMP Redirect
 - 2.1.3.1.5 Abusing ICMP Redirect
 - 2.2 Analyzing Common Application Protocol Traffic & Attacks
 - 2.2 Analyzing Common Application Protocol Traffic & At...
- ▼ 2.2.1 Microsoft-specific Protocols

2.2.1.1 NetBIOS

Network Basic Input/Output System (NetBIOS) is a set of protocols developed in the early 1980s for LAN communications, in order to provide services for the session layer (fifth layer in the OSI model).

A few years later, it was adopted by Microsoft for their networking over LAN, and then it was migrated for working over TCP/IP (NetBIOS over TCP/IP—NBT), which is discussed in RFC 1001 and 1002.

OUTLINE

CONTENTS

- 2.1.3.1.5 Abusing ICMP Redirect
- 2.1.3.1.5 Abusing ICMP Redirect
- 2.1.3.1.5 Abusing ICMP Redirect
- 2.1.3.1.5 Abusing ICMP Redirect
- 2.1.3.1.5 Abusing ICMP Redirect

2.1 Analyzing & Detecting Transport Layer Attacks

2.2 Analyzing Common Application Protocol Traffic & Attacks

2.2 Analyzing Common Application Protocol Traffic & At...

2.2.1 Microsoft-specific Protocols

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

In today's networks, NetBIOS provides three services:

- Name service (port 137) for name registration and name to IP address resolution. Also referred to as NetBIOS-NS.
- Datagram distribution service (port 138) for service announcements by clients and servers. Also referred to as NetBIOS-DGM.
- Session service (port 139) for session negotiation between hosts. This is used for accessing files, opening directories, and so on. Also referred to as NetBIOS-SSN.

OUTLINE

- 2.1 Analyzing & Detecting Transport Layer Attacks
 - 2.1.3.1.5 Abusing ICMP Redirect
 - 2.1.3.1.5 Abusing ICMP Redirect
 - 2.1.3.1.5 Abusing ICMP Redirect
 - 2.1.3.1.5 Abusing ICMP Redirect
 - 2.1.3.1.5 Abusing ICMP Redirect
- 2.2 Analyzing Common Application Protocol Traffic & Attacks
 - 2.2 Analyzing Common Application Protocol Traffic & Attacks
 - 2.2.1 Microsoft-specific Protocols
 - 2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

NBNS is the service that registers and translates names to IP addresses. The Microsoft environment was implemented with WINS, and as most networks did not use it, it was later replaced by DNS. It works over UDP port 137.

OUTLINE

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1 Analyzing & Detecting Transport Layer Attacks

▼ 2.2 Analyzing Common Application Protocol Traffic & Attacks

2.2 Analyzing Common Application Protocol Traffic & At...

▼ 2.2.1 Microsoft-specific Protocols

▼ 2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

NBDS is used for service announcements by clients and servers. With this service, devices on the network announce their names, services that they can provide to other devices on the network, and how to connect to these services. It works over UDP port 138

OUTLINE

- 2.1 Analyzing & Detecting Transport Layer Attacks
 - 2.1.3.1.5 Abusing ICMP Redirect
 - 2.1.3.1.5 Abusing ICMP Redirect
 - 2.1.3.1.5 Abusing ICMP Redirect
- 2.2 Analyzing Common Application Protocol Traffic & Attacks
 - 2.2 Analyzing Common Application Protocol Traffic & Attacks
 - 2.2.1 Microsoft-specific Protocols
 - 2.2.1.1 NetBIOS
 - 2.2.1.1 NetBIOS
 - 2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

NetBIOS is used to establish sessions between hosts, open or save files, and execute remote files and other sessions over the network. It works over TCP port 139.

OUTLINE

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1 Analyzing & Detecting Transport Layer Attacks

2.2 Analyzing Common Application Protocol Traffic & Attacks

2.2 Analyzing Common Application Protocol Traffic & Attacks

2.2.1 Microsoft-specific Protocols

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

There are additional protocols, such as **Server Message Block (SMB)**, that run over NBSS for transaction operations and over NBDS for service announcement, spools for printer requests, and several others.

OUTLINE

2.1.3.1.5 Abusing ICMP Redirect

2.1.3.1.5 Abusing ICMP Redirect

2.1 Analyzing & Detecting Transport Layer Attacks

▼ 2.2 Analyzing Common Application Protocol Traffic & Attacks

2.2 Analyzing Common Application Protocol Traffic & Attacks

▼ 2.2.1 Microsoft-specific Protocols

▼ 2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.2 SMB

SMB is a protocol that is used for browsing directories, copying files, accessing services such as printers, and several other operations over the network.

Common Internet File System (CIFS) is a form, or flavor, of SMB.

OUTLINE

2.1 Analyzing & Detecting
Transport Layer Attacks

▼ 2.2 Analyzing Common Application
Protocol Traffic & Attacks

2.2 Analyzing Common
Application Protocol Traffic & At...

▼ 2.2.1 Microsoft-specific Protocols

▼ 2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

▼ 2.2.1.2 SMB

2.2.1.2 SMB

SMB runs on top of the session layer protocols such as NetBIOS as originally designed or can also run directly over TCP port 445.

OUTLINE

Transport Layer Protocols

▼ 2.2 Analyzing Common Application Protocol Traffic & Attacks

2.2 Analyzing Common Application Protocol Traffic & At...

▼ 2.2.1 Microsoft-specific Protocols

▼ 2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

▼ 2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

SMB 2.0 was introduced by Microsoft in 2006 in Windows Vista, with the intention of reducing the commands and subcommands required in the SMB 1.0 protocol.

Even though SMB 2.0 came out as a proprietary protocol, Microsoft published the standard to allow other systems to interoperate with their operating systems

OUTLINE

Protocol Home > Servers

2.2 Analyzing Common
Application Protocol Traffic & At...

▼ 2.2.1 Microsoft-specific Protocols

▼ 2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

▼ 2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

SMB 2.1 was released with Windows 7 and Server 2008 R2 with performance improvements compared to SMB 2.0.

OUTLINE

Application Protocol Framework

▼ 2.2.1 Microsoft-specific Protocols

▼ 2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

▼ 2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

SMB 3.0 (earlier referred to as SMB 2.2) was introduced with Windows 8 and Server 2012. SMB 3.0 has significant performance improvements (compared to earlier releases) in order to support virtualization occurring in the data center computing environment.

OUTLINE

▼ 2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

▼ 2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

Detection

SMB works in a client-server model, where the client makes specific requests to the server, and based on the requests, the server responds accordingly. Most of the requests are related to accessing filesystems, while other forms of requests involve Inter-Process Communication (IPC).

Code 0 means STATUS_OK, which implies that everything works fine and there is no problem. Any other code should be examined.

```
14550 10.1.70.95 203 SMB 93 NT Create AndX Response, FID: 0x0000, Error: STATUS_ACCESS_DENIED
14551 203 10.1.70.95 SMB 146 NT Create AndX Request, Path: \
14552 10.1.70.95 203 SMB 93 NT Create AndX Response, FID: 0x0000, Error: STATUS_ACCESS_DENIED

Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: nts-den (4132), Seq: 330825, Ack: 213
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Header
  Server Component: SMB
  [Response to: 14547]
  [Time from request: 0.019610000 seconds]
  SMB Command: NT Create AndX (0xa2)
  NT Status: STATUS_ACCESS_DENIED (0xc0000022)
  Flags: 0x98
  Flags2: 0xc803
  Process ID High: 0
  Signature: 0000000000000000
  Reserved: 0000
  Tree ID: 8199 (\\NAS01\HOMEDIR)
  Process ID: 1544
  User ID: 14339
  Multiplex ID: 46595
  NT Create AndX Response (0xa2)
```

STATUS_ACCESS_DENIED SMB error example

OUTLINE

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

▼ 2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

Detection

SMB works in a client-server model, where the client makes specific requests to the server, and based on the requests, the server responds accordingly. Most of the requests are related to accessing filesystems, while other forms of requests involve Inter-Process Communication (IPC).

Code 0 means STATUS_OK, which implies that everything works fine and there is no problem. Any other code should be examined.

```
203.12.106.10 SMB 93 NT Trans Response, FID: 0x0001, NT NOTIFY, Error: STATUS_CANCELLED
203.12.106.10 SMB 478 Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCESSING_REQUIRED
203.12.106.10 SMB 93 Tree Connect AndX Response, Error: STATUS_ACCESS_DENIED
203.12.106.10 SMB 478 Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCESSING_REQUIRED
203.12.106.10 SMB 93 Tree Connect AndX Response, Error: STATUS_ACCESS_DENIED
203.12.106.10 SMB 478 Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCESSING_REQUIRED
203.12.106.10 SMB 93 Tree Connect AndX Response, Error: STATUS_ACCESS_DENIED
203.12.106.14 SMB 93 Trans2 Response, GET_DFS_REFERRAL, Error: STATUS_NO_SUCH_DEVICE
203.12.106.14 SMB 93 NT Create AndX Response, FID: 0x0000, Error: STATUS_ACCESS_DENIED
```

SMB errors you should examine

```
21 // Check the result of the request
22 if (NtStatus == STATUS_OK)
23     ReportStatusOk();
24 else
25 {
26     // Check what the result is based on the error code
27     if (NtStatus == STATUS_ACCESS_DENIED)
28     {
29         ReportAccessDenied();
30     }
31     // ...
32 }
```

OUTLINE

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

▼ 2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

Detection

Certain versions of Windows allowed for anonymous and passwordless authentication. Such a connection is often referred to as a NULL session. A NULL session may be limited in terms of privileges, but it could be used to execute various RPC calls and subsequently perform information gathering or user enumeration.

On your right, you see traffic generated by Nmap's *smb-enum-users* script. Notice the **successful** attempt to identify if a NULL session is possible. Also notice the RPC calls inside the subsequent rectangles, that were run to provide the attacker with a user list.

```
TCP 57500 -> 445 [SYN] Seq=8 Win=2280 Len=0 MSS=1460 SACK_PERM=1 TSval=909964135 TSecr=
TCP 445 -> 57500 [SYN, ACK] Seq=8 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 TSva
TCP 57500 -> 445 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=909964135 TSecr=1641308
SMB Negotiate Protocol Request
SMB Negotiate Protocol Response
TCP 57500 -> 445 [ACK] Seq=54 Ack=119 Win=29312 Len=0 TSval=909964135 TSecr=1641308
SMB Session Setup AndX Request, User: anonymous
SMB Session Setup AndX Response
SMB Tree Connect AndX Request, Path: \\192.168.57.105\IPC$
SMB Tree Connect AndX Response
SMB NT Create AndX Request, FID: 0x4000, Path: \\smb
SMB NT Create AndX Response, FID: 0x4000
DCERPC Bind: call_id: 1004795585, Fragment: Single, 1 context items: SAMR V1.0 (32bit MD
SMB Write AndX Request, FID: 0x4000, 72 bytes
SMB Read AndX Request, FID: 0x4000, 2048 bytes at offset 0
DCERPC Bind_ack: call_id: 1004795585, Fragment: Single, max_xmit: 2048 max_recv: 2048, 1
SMB Connect4 request
SMB Write AndX Response, FID: 0x4000, 84 bytes
SMB Read AndX Request, FID: 0x4000, 4097 bytes at offset 0
SMB Connect4 response
SMB EnumDomains request
SMB Write AndX Response, FID: 0x4000, 52 bytes
SMB Read AndX Request, FID: 0x4000, 4097 bytes at offset 0
SMB EnumDomains response
SMB LOOKUPDOMAIN request, User-PC[Long frame (2 bytes)]
SMB Write AndX Response, FID: 0x4000, 80 bytes
SMB Read AndX Request, FID: 0x4000, 4097 bytes at offset 0
SMB LookupDomain response
SMB OpenDomain request
SMB Write AndX Response, FID: 0x4000, 76 bytes
SMB Read AndX Request, FID: 0x4000, 4097 bytes at offset 0
SMB OpenDomain response
SMB QueryDisplayInfo request
SMB Write AndX Response, FID: 0x4000, 80 bytes
SMB Read AndX Request, FID: 0x4000, 4097 bytes at offset 0
SMB QueryDisplayInfo response
SMB Close request
SMB Write AndX Response, FID: 0x4000, 44 bytes
SMB Read AndX Request, FID: 0x4000, 4097 bytes at offset 0
SMB Close response
SMB Close request
SMB Write AndX Response, FID: 0x4000, 44 bytes
SMB Read AndX Request, FID: 0x4000, 4097 bytes at offset 0
SMB Close response
SMB Tree Disconnect Request
SMB Tree Disconnect Response
SMB Logoff AndX Request
SMB Logoff AndX Response
TCP 57500 -> 445 [FIN, ACK] Seq=1043 Ack=1507 Win=35328 Len=0 TSval=909964141 TSecr=164
```

OUTLINE

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

▼ 2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.3 MSRPC

The RPC (Remote Procedure Call) mechanism allows an application to seamlessly invoke remote procedures, as if these procedures were executed locally.

There are two main implementations of the RPC mechanism:

- ONC RPC
- DCE RPC

OUTLINE

2.2.1.1 NetBIOS

2.2.1.1 NetBIOS

▼ 2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

▼ 2.2.1.3 MSRPC

2.2.1.3 MSRPC

MSRPC is the Microsoft implementation of the DCE RPC mechanism. In particular, Microsoft added new transport protocols for DCE RPC, which use named pipes carried into the SMB protocol.

File operations utilize SMB/CIFS, whereas administrative operations, resource management operations etc. utilize MSRPC.

OUTLINE

2.2.1.1 NetBIOS

▼ 2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

▼ 2.2.1.3 MSRPC

2.2.1.3 MSRPC

2.2.1.3 MSRPC

There are multiple MSRPC implementations:

- RCP over SMB ([this](#) is an example RPC over SMB traffic)
- DCOM (RPC directly over TCP/UDP) [TCP/UDP port 135]
- RPC over HTTP or HTTPS [TCP/UDP port 593]

The RPC Endpoint Mapper comes into play in those implementations

OUTLINE

▼ 2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

▼ 2.2.1.3 MSRPC

2.2.1.3 MSRPC

▼ 2.2.1.3 MSRPC

2.2.1.3.1 MSRPC Over TCP Session

There are multiple MSRPC implementations:

- RCP over SMB ([this](#) is an example RPC over SMB traffic)
- **DCOM** (RPC directly over TCP/UDP) [TCP/UDP port 135]
- RPC over HTTP or HTTPS [TCP/UDP port 593]

The RPC Endpoint Mapper comes into play in those implementations

OUTLINE

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

▼ 2.2.1.3 MSRPC

2.2.1.3 MSRPC

▼ 2.2.1.3 MSRPC

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC Over TCP Session

TCP/IP RPC services listen on dynamic TCP or UDP ports. Thus, to reach a given RPC service, identified by its interface identifier (UUID), a port mapping service is necessary.

The portmapper service is an RPC service listening on different endpoints.

OUTLINE

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

▼ 2.2.1.3 MSRPC

2.2.1.3 MSRPC

▼ 2.2.1.3 MSRPC

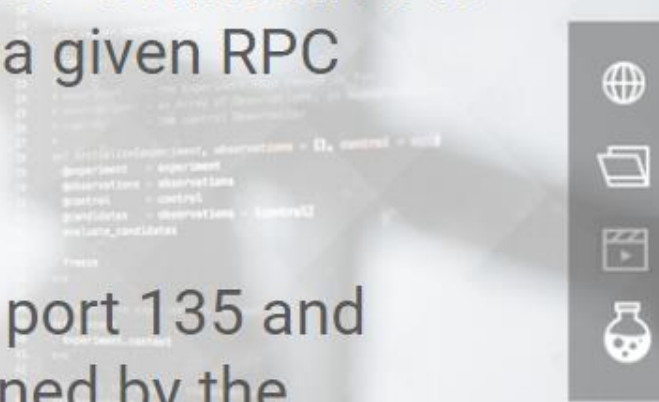
2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC Over TCP Session

Typically, to discover the port on which a given RPC service can be reached, a client will establish a TCP connection to port 135, asking for the port allocated to a given RPC service.

Then, the client closes the connection to port 135 and opens a new connection to the port returned by the portmapper service.



OUTLINE

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

▼ 2.2.1.3 MSRPC

2.2.1.3 MSRPC

▼ 2.2.1.3 MSRPC

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC Over TCP Session

Consider the *schannel_win7-member_to_win2k3-dc.pcap* file found in this module's resources, to see how the MSRPC over TCP session implementation looks like on the wire.

Note: Study the pcap file from packet #129 onwards.

OUTLINE

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

▼ 2.2.1.3 MSRPC

2.2.1.3 MSRPC

▼ 2.2.1.3 MSRPC

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.2 HTTP(S)

Everyday, we browse numerous websites. While doing so, we are generating HTTP(S) traffic.

It is time we give the HTTP and HTTPS protocols a better look...

OUTLINE

2.2.1.2 SMB

2.2.1.2 SMB

2.2.1.2 SMB

▼ 2.2.1.3 MSRPC

2.2.1.3 MSRPC

▼ 2.2.1.3 MSRPC

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

▼ 2.2.2 HTTP(S)

2.2.2.1 HTTP

A few things to remember about HTTP:

- HTTP traffic consists of a series of requests and responses known as **messages**.
 - Client will make a request & server will respond.
 - HTTP responses include a **3 digit status code**.
- HTTP messages include a **message header** and **body**.
- HTTP uses **methods** to perform various operations.
 - 8 HTTP methods defined in [RFC 2616](https://www.ietf.org/rfc/rfc2616.txt).
 - Not all methods will be permitted by web server.

OUTLINE

2.2.1.2 SMB

2.2.1.2 SMB

▼ 2.2.1.3 MSRPC

2.2.1.3 MSRPC

▼ 2.2.1.3 MSRPC

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

▼ 2.2.2 HTTP(S)

▼ 2.2.2.1 HTTP

2.2.2.1 HTTP

Some facts to help distinguish normal and suspicious HTTP traffic

Normal HTTP Traffic	Suspicious HTTP Traffic
Port 80, TCP Port 8080, TCP (used as alternate) Port 8088, TCP (used as alternate)	Malicious binaries (backdoors), scripts, web shells, etc. will use this port because typically in all corporate environments the port is open.
Plaintext traffic	If the traffic is encrypted then most likely it's being used for malicious traffic. Malicious traffic can be in plaintext as well.
Web server typically in FQDN format.	Server will point to an IP address instead of FQDN format.

OUTLINE

2.2.1.2 SMB

▼ 2.2.1.3 MSRPC

2.2.1.3 MSRPC

▼ 2.2.1.3 MSRPC

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

▼ 2.2.2 HTTP(S)

▼ 2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

Let's now see how normal HTTP traffic looks like.

OUTLINE

▼ 2.2.1.3 MSRPC

2.2.1.3 MSRPC

▼ 2.2.1.3 MSRPC

2.2.1.3.1 MSRPC Over TCP Session

2.2.1.3.1 MSRPC Over TCP Session

2.2.1.3.1 MSRPC Over TCP Session

2.2.1.3.1 MSRPC Over TCP Session

▼ 2.2.2 HTTP(S)

▼ 2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

No.	Time	Source	Destination	Protocol	Length	Info
3	0.0508820	10.54.15...	10.54.15...	TCP	74	50982 → 80 [SYN, Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2580544 TSecr=0 WS=128]
5	0.0965761	10.54.15...	10.54.15...	TCP	74	80 → 50982 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=4294938685 TSecr=2580544 WS=4
6	0.0966111	10.54.15...	10.54.15...	TCP	66	50982 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2580568 TSecr=4294938685
7	0.0968141	10.54.15...	10.54.15...	HTTP	347	GET / HTTP/1.1
8	0.1469519	10.54.15...	10.54.15...	TCP	66	80 → 50982 [ACK] Seq=1 Ack=282 Win=15552 Len=0 TSval=4294938698 TSecr=2580568
9	0.1809993	10.54.15...	10.54.15...	HTTP	654	HTTP/1.1 200 OK (text/html)

Here is a brief summary concerning the image above:

- We are seeing 6 packets (4 relating to TCP and 2 relating to HTTP).
- Packets 3-6 is the **TCP Handshake**. HTTP relies on TCP for reliability.
- Packet 7 we notice a **HTTP method** (GET).
- Packet 9 we notice a **HTTP response code** (200 OK).

OUTLINE

2.2.1.3 MSRPC

▼ 2.2.1.3 MSRPC

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

▼ 2.2.2 HTTP(5)

▼ 2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

No.	Time	Source	Destination	Protocol	Length	Info
3	0.0508829...	10.54.15...	10.54.15...	TCP	74	50982 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2580544 TSecr=0 WS=128
5	0.0965761...	10.54.15...	10.54.15...	TCP	74	80 → 50982 [SYN, ACK] Seq=0 Ack=1 Win=14490 Len=0 MSS=1337 SACK_PERM=1 TSval=4294938685 TSecr=2580544 WS=4
6	0.0966111...	10.54.15...	10.54.15...	TCP	66	50982 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2580568 TSecr=4294938685
7	0.0968141...	10.54.15...	10.54.15...	HTTP	347	GET / HTTP/1.1
8	0.1469519...	10.54.15...	10.54.15...	TCP	66	80 → 50982 [ACK] Seq=1 Ack=262 Win=15552 Len=0 TSval=4294938690 TSecr=2580568
9	0.1809993...	10.54.15...	10.54.15...	HTTP	654	HTTP/1.1 200 OK (text/html)

▶ Frame 7: 347 bytes on wire (2776 bits), 347 bytes captured (2776 bits) on interface 0
▶ Ethernet II, Src: 26:11:59:88:53:02 (26:11:59:88:53:02), Dst: vmware_a1:61:06 (00:50:56:a1:61:06)
▶ Internet Protocol Version 4, Src: 10.54.15.100, Dst: 10.54.15.68
▶ Transmission Control Protocol, Src Port: 50982, Dst Port: 80, Seq: 1, Ack: 1, Len: 281
▶ Hypertext Transfer Protocol

In the image above we see the packet information as it pertains the different layers of the OSI model.

- Ethernet II = MAC Addresses (SRC, DST)
- Internet Protocol Version 4 = IP Address (SRC, DST)
- Transmission Control Protocol = TCP (SRC PORT, DST PORT)
- Hypertext Transfer Protocol = HTTP

OUTLINE

▼ 2.2.1.3 MSRPC

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

▼ 2.2.2 HTTP(S)

▼ 2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

No.	Time	Source	Destination	Protocol	Length	Info
3	0.0508829	10.54.15...	10.54.15...	TCP	7	80 → 50982 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2580544 TSecr=0 WS=128
5	0.0965761	10.54.15...	10.54.15...	TCP	7	80 → 50982 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=4294938685 TSecr=2580544 WS=4
6	0.0966111	10.54.15...	10.54.15...	TCP	6	50982 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2580568 TSecr=4294938685
7	0.0968141	10.54.15...	10.54.15...	HTTP	34	GET / HTTP/1.1
8	0.1469519	10.54.15...	10.54.15...	TCP	6	80 → 50982 [ACK] Seq=1 Ack=282 Win=15552 Len=0 TSval=4294938698 TSecr=2580568
9	0.1809993	10.54.15...	10.54.15...	HTTP	654	HTTP/1.1 200 OK (text/html)

▶ Frame 7: 347 bytes on wire (2776 bits), 347 bytes captured (2776 bits) on interface 0
▶ Ethernet II, Src: 26:11:59:88:53:02 (26:11:59:88:53:02), Dst: vmware_a1:61:66 (00:50:56:a1:61:66)
▶ Internet Protocol Version 4, Src: 10.54.15.100, Dst: 10.54.15.60
▶ Transmission Control Protocol, Src Port: 50982, Dst Port: 80, Seq: 1, Ack: 1, Len: 281
▶ Hypertext Transfer Protocol

So far we can tell that the traffic is using an expected port, port 80. Remember that this port doesn't necessarily have to be port 80. An organization can decide to use a different port altogether but we know the standard port is 80 and if it's port 80 then we're expecting web traffic.

OUTLINE

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

▼ 2.2.2 HTTP(S)

▼ 2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

```
▶ Frame 7: 347 bytes on wire (2776 bits), 347 bytes captured (2776 bits) on interface 0
▶ Ethernet II, Src: 26:11:59:88:53:02 (26:11:59:88:53:02), Dst: vmware_a1:61:66 (00:50:56:a1:61:66)
▶ Internet Protocol Version 4, Src: 10.54.15.100, Dst: 10.54.15.68
▶ Transmission Control Protocol, Src Port: 50982, Dst Port: 80, Seq: 1, Ack: 1, Len: 281
▶ Hypertext Transfer Protocol
  ▶ GET / HTTP/1.1\r\n
    Host: 10.54.15.68\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
    \r\n
    [Full request URI: http://10.54.15.68/]
    [HTTP request 1/2]
    [Response in frame: 9]
    [Next request in frame: 11]
```

Here, we're looking at the HTTP section of the packet details. The HTTP method is **GET** and the host is **not** in FQDN format. The *http://IP* format should alarm you, but in this case its pointing to an internal server and that is normal behavior in corporate environments. The source host has an IP of 10.54.15.100.

If we double click the link, **[Response in frame: 9]**, in the HTTP section of the packet details of packet 7 we'll jump to packet 9 which contains the packet details of the **HTTP response** to the **HTTP request**.

OUTLINE

OVER TCP SESSION

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

▼ 2.2.2 HTTP(S)

▼ 2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP



```
Frame 9: 654 bytes on wire (5232 bits), 654 bytes captured (5232 bits) on interface 0
Ethernet II, Src: vmware_a1:f4:d0 (00:50:56:a1:f4:d0), Dst: 26:11:59:88:53:02 (26:11:59:88:53:02)
Internet Protocol Version 4, Src: 10.54.15.68, Dst: 10.54.15.100
Transmission Control Protocol, Src Port: 80, Dst Port: 50982, Seq: 1, Ack: 282, Len: 588
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    Date: Tue, 23 May 2017 17:23:14 GMT\r\n
    Server: Apache/2.2.22 (Debian)\r\n
    X-Powered-By: PHP/5.4.4-14+deb7u14\r\n
    Vary: Accept-Encoding\r\n
    Content-Encoding: gzip\r\n
    Content-Length: 315\r\n
    Keep-Alive: timeout=5, max=100\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/html\r\n
    \r\n
    [HTTP response 1/2]
    [Time since request: 0.084185252 seconds]
    [Request in frame: 7]
    [Next request in frame: 11]
    [Next response in frame: 12]
    Content-encoded entity body (gzip): 315 bytes -> 546 bytes
    File Data: 546 bytes
  Line-based text data: text/html
    <!DOCTYPE html>\n
    <html>\n
```

Within this packet we can do the same, jump back to packet 7 using the link **[Request in frame: 7]**.

OUTLINE

OVERVIEW SESSION

2.2.1.3.1 MSRPC
Over TCP Session

2.2.1.3.1 MSRPC
Over TCP Session

▼ 2.2.2 HTTP(S)

▼ 2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

```
Line-based text data: text/html
<!DOCTYPE html>\n
<html>\n
  <head>\n
    <meta charset="UTF-8">\n
    <title>Log-in - login.labs</title>\n
    <link rel="stylesheet" href="css/style.css" media="screen" type="text/css" />\n
  </head>\n
  <body>\n
    <div class="login-card">\n
      <h1>Log-in</h1><br>\n
      <form method="post" action="login.php">\n
        <input type="text" name="user" autocomplete="off" placeholder="Username">\n
        <input type="password" name="pass" placeholder="Password">\n
        <input type="submit" name="login" class="login login-submit" value="login">\n
      </form>\n
    </div>\n
  </body>\n
</html>
```

Above we notice a section called **Line-based text data: text/html** within the packet details. We can clearly see that the packet is returning a login page for the server.

OUTLINE

OVER THE SESSION

2.2.1.3.1 MSRPC
Over TCP Session

▼ 2.2.2 HTTP(S)

▼ 2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

The same information can be obtained by right-clicking packet #7 or packet #9 and selecting **Follow -> Follow TCP Stream**.

Here we see the same details but only information from the packet that pertains to HTTP.

Now remember that HTTP traffic is supposed to be in cleartext, which we have seen so far. Let's also look at the packet stream after logging into the internal server to see credentials in cleartext.



```
GET / HTTP/1.1
Host: 10.54.15.68
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive

HTTP/1.1 200 OK
Date: Tue, 23 May 2017 17:29:40 GMT
Server: Apache/2.2.22 (Debian)
X-Powered-By: PHP/5.4.4-14+deb7u14
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 315
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

...
u..A..1...#..9p..@N...;..I...{A..?..=..?..w..X..E...c..J...
0..08...h...h...ep..8...{...93...R...3...J...B..9A...vAk781D= $...f...W..E...
7...C...}...
#...j2..2..4../...t...
At..B...x..g...
3 client pkts, 3 server pkts, 3 bytes
Entire conversation (869 bytes)
Show and save data as ASCII
Stream 0
Find:
Find Next
Filter Out This Stream
Print
Save as...
Back
Close
Help
```

OUTLINE

OVERVIEW SESSION

▼ 2.2.2 HTTP(S)

▼ 2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

Before looking at cleartext credentials let's test our packet analysis skills.
How many TCP streams are there in the packet capture below?

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	26:11:59:88:53:02	Broadcast	ARP	42	who has 10.54.15.68? Tell 10.54.15.100
2	0.05084186	vmware_81:61:06	26:11:59:88:53:02	ARP	60	10.54.15.68 is at 00:50:56:a1:61:06
3	0.050882951	10.54.15.100	10.54.15.68	TCP	74	50982 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2580544 TSecr=0 WS=128
4	0.050900684	vmware_81:f4:d0	26:11:59:88:53:02	ARP	60	10.54.15.68 is at 00:50:56:a1:f4:d0
5	0.096570317	10.54.15.68	10.54.15.100	TCP	74	80 → 50982 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=4294938685 TSecr=2580544 WS=4
6	0.096611102	10.54.15.100	10.54.15.68	TCP	66	50982 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2580568 TSecr=4294938685
7	0.096614120	10.54.15.100	10.54.15.68	HTTP	347	GET / HTTP/1.1
8	0.146951934	10.54.15.68	10.54.15.100	TCP	66	80 → 50982 [ACK] Seq=1 Ack=282 Win=15552 Len=0 TSval=4294938686 TSecr=2580568
9	0.180999272	10.54.15.68	10.54.15.100	HTTP	654	HTTP/1.1 200 OK (text/html)
10	0.181016327	10.54.15.100	10.54.15.68	TCP	66	50982 → 80 [ACK] Seq=282 Ack=589 Win=30464 Len=0 TSval=2580589 TSecr=4294938706
11	0.212109429	10.54.15.100	10.54.15.68	HTTP	345	GET /css/style.css HTTP/1.1
12	0.286521761	10.54.15.68	10.54.15.100	HTTP	1242	HTTP/1.1 200 OK (text/css)
13	0.327365188	10.54.15.100	10.54.15.68	TCP	66	50982 → 80 [ACK] Seq=561 Ack=1765 Win=33280 Len=0 TSval=2580626 TSecr=4294938732
14	0.290026896	10.54.15.68	10.54.15.100	TCP	66	80 → 50982 [FIN, ACK] Seq=1765 Ack=561 Win=16624 Len=0 TSval=4294939983 TSecr=2580626
15	0.290130806	10.54.15.100	10.54.15.68	TCP	66	50982 → 80 [FIN, ACK] Seq=561 Ack=1766 Win=33280 Len=0 TSval=2581866 TSecr=4294939983
16	0.335960497	10.54.15.68	10.54.15.100	TCP	66	80 → 50982 [ACK] Seq=1766 Ack=562 Win=16624 Len=0 TSval=4294939995 TSecr=2581866
17	18.179909185	fe80::2411:59ff:fe80::ff02::2	ff02::2	ICMPv6	70	Router Solicitation from 26:11:59:88:53:02
18	20.062049221	10.54.15.100	10.54.15.68	TCP	74	51008 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2580759 TSecr=0 WS=128
19	20.067960258	10.54.15.68	10.54.15.100	TCP	74	80 → 51008 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=4294943088 TSecr=2580759 WS=4
20	20.067980940	10.54.15.100	10.54.15.68	TCP	66	51008 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2580771 TSecr=4294943088
21	20.068172432	10.54.15.100	10.54.15.68	HTTP	350	GET /favicon.ico HTTP/1.1
22	20.065020739	10.54.15.68	10.54.15.100	TCP	66	80 → 51008 [ACK] Seq=1 Ack=293 Win=15552 Len=0 TSval=4294943900 TSecr=2580771
23	20.057982825	10.54.15.68	10.54.15.100	HTTP	568	HTTP/1.1 404 Not Found (text/html)
24	20.057977826	10.54.15.100	10.54.15.68	TCP	66	51008 → 80 [ACK] Seq=293 Ack=503 Win=30336 Len=0 TSval=2580783 TSecr=4294943900
25	25.062682870	10.54.15.68	10.54.15.100	TCP	66	80 → 51008 [FIN, ACK] Seq=503 Ack=293 Win=15552 Len=0 TSval=4294945152 TSecr=2580783
26	25.062789254	10.54.15.100	10.54.15.68	TCP	66	51008 → 80 [FIN, ACK] Seq=293 Ack=504 Win=30336 Len=0 TSval=2587034 TSecr=4294945152
27	26.008812475	10.54.15.68	10.54.15.100	TCP	66	80 → 51008 [ACK] Seq=504 Ack=294 Win=15552 Len=0 TSval=4294945163 TSecr=2587034
28	42.017326020	10.54.15.100	10.54.15.68	TCP	74	51012 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2591273 TSecr=0 WS=128
29	42.069116809	10.54.15.68	10.54.15.100	TCP	74	80 → 51012 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=4294949402 TSecr=2591273 WS=4
30	42.069140602	10.54.15.100	10.54.15.68	TCP	66	51012 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2591286 TSecr=4294949402
31	42.069353931	10.54.15.100	10.54.15.68	HTTP	501	POST /login.php HTTP/1.1 (application/x-www-form-urlencoded)
32	43.018754696	10.54.15.68	10.54.15.100	TCP	66	80 → 51012 [ACK] Seq=1 Ack=436 Win=19952 Len=0 TSval=4294949415 TSecr=2591286
33	43.033910817	10.54.15.68	10.54.15.100	HTTP	604	HTTP/1.1 200 OK (text/html)
34	43.033823724	10.54.15.100	10.54.15.68	TCP	66	51012 → 80 [ACK] Seq=436 Ack=539 Win=30336 Len=0 TSval=2591302 TSecr=4294949420
35	48.040277295	10.54.15.68	10.54.15.100	TCP	66	80 → 51012 [FIN, ACK] Seq=539 Ack=436 Win=15552 Len=0 TSval=4294950671 TSecr=2591302
36	48.040506972	10.54.15.100	10.54.15.68	TCP	66	51012 → 80 [FIN, ACK] Seq=436 Ack=540 Win=30336 Len=0 TSval=2592554 TSecr=4294950671
37	48.089829220	10.54.15.68	10.54.15.100	TCP	66	80 → 51012 [ACK] Seq=540 Ack=437 Win=15552 Len=0 TSval=4294950683 TSecr=2592554
38	91.063620400	fe80::2411:59ff:fe80::ff02::2	ff02::2	ICMPv6	70	Router Solicitation from 26:11:59:88:53:02

OUTLINE

▼ 2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

There are a couple ways to determine this within the Wireshark GUI:

1. Look at the TCP packets then look at source ports and destination ports.
2. Use display filters and increment the numbers to show a particular TCP stream until nothing is displayed.
3. Select **Statistics -> Conversations** and then click the **TCP tab**.
4. In the Follow TCP Stream window, at the bottom right, you can navigate through the different streams that Wireshark detected.

OUTLINE

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

1. From the packet capture we can see different source ports, so, each different source port will have its own TCP Handshake followed by its corresponding traffic.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	26:11:59:88:53:02	Broadcast	ARP	42	who has 10.54.15.68? Tell 10.54.15.100
2	0.050804109	vmware_a1:61:06	26:11:59:88:53:02	ARP	60	10.54.15.68 is at 00:50:56:a1:61:06
3	0.0508082951	10.54.15.100	10.54.15.68	TCP	74	50982 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2580544 TSecr=0 WS=128
4	0.050906894	vmware_a1:f4:d0	26:11:59:88:53:02	ARP	60	10.54.15.68 is at 00:50:56:a1:f4:d0
5	0.090576117	10.54.15.68	10.54.15.100	TCP	74	80 → 50982 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=4294938685 TSecr=2580544 WS=4
6	0.090611102	10.54.15.100	10.54.15.68	TCP	66	50982 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2580568 TSecr=4294938685
7	0.090814120	10.54.15.100	10.54.15.68	HTTP	347	GET / HTTP/1.1
8	0.146951934	10.54.15.68	10.54.15.100	TCP	66	80 → 50982 [ACK] Seq=1 Ack=282 Win=15552 Len=0 TSval=4294938698 TSecr=2580568
9	0.180990372	10.54.15.68	10.54.15.100	HTTP	654	HTTP/1.1 200 OK (text/html)
10	0.181916327	10.54.15.100	10.54.15.68	TCP	66	50982 → 80 [ACK] Seq=282 Ack=589 Win=38484 Len=0 TSval=2580589 TSecr=4294938706
11	0.212109429	10.54.15.100	10.54.15.68	HTTP	345	GET /css/style.css HTTP/1.1
12	0.266521761	10.54.15.68	10.54.15.100	HTTP	1242	HTTP/1.1 200 OK (text/css)
13	0.327365188	10.54.15.100	10.54.15.68	TCP	66	50982 → 80 [ACK] Seq=561 Ack=1765 Win=33280 Len=0 TSval=2580626 TSecr=4294938732
14	0.290626896	10.54.15.68	10.54.15.100	TCP	66	80 → 50982 [FIN, ACK] Seq=1765 Ack=561 Win=10624 Len=0 TSval=4294939983 TSecr=2580626
15	0.290130609	10.54.15.100	10.54.15.68	TCP	66	50982 → 80 [FIN, ACK] Seq=561 Ack=1766 Win=33280 Len=0 TSval=2581066 TSecr=4294939983
16	0.335860407	10.54.15.68	10.54.15.100	TCP	66	80 → 50982 [ACK] Seq=1766 Ack=562 Win=16924 Len=0 TSval=4294939995 TSecr=2581066
17	18.175908185	fe80::2411:59ff:fe8...ff02::2		ICMPv6	70	Router Solicitation from 26:11:59:88:53:02
18	20.062048221	10.54.15.100	10.54.15.68	TCP	74	80 → 51012 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2585759 TSecr=0 WS=128
19	20.067960258	10.54.15.68	10.54.15.100	TCP	74	80 → 51012 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=4294943888 TSecr=2585759 WS=4
20	20.067988940	10.54.15.100	10.54.15.68	TCP	66	51008 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2585771 TSecr=4294943888
21	20.068172432	10.54.15.100	10.54.15.68	HTTP	358	GET /favicon.ico HTTP/1.1
22	20.068920739	10.54.15.68	10.54.15.100	TCP	66	80 → 51008 [ACK] Seq=1 Ack=293 Win=15552 Len=0 TSval=4294943900 TSecr=2585771
23	20.067862825	10.54.15.68	10.54.15.100	HTTP	568	HTTP/1.1 404 Not Found (text/html)
24	20.067877026	10.54.15.100	10.54.15.68	TCP	66	51008 → 80 [ACK] Seq=293 Ack=503 Win=38336 Len=0 TSval=2585783 TSecr=4294943900
25	25.062682870	10.54.15.68	10.54.15.100	TCP	66	80 → 51008 [FIN, ACK] Seq=503 Ack=293 Win=15552 Len=0 TSval=4294945152 TSecr=2585783
26	25.062789254	10.54.15.100	10.54.15.68	TCP	66	51008 → 80 [FIN, ACK] Seq=293 Ack=504 Win=38336 Len=0 TSval=2587954 TSecr=4294945152
27	26.000612475	10.54.15.68	10.54.15.100	TCP	66	80 → 51008 [ACK] Seq=504 Ack=294 Win=15552 Len=0 TSval=4294945163 TSecr=2587954
28	29.012091020	10.54.15.100	10.54.15.68	TCP	74	80 → 51012 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2591273 TSecr=0 WS=128
29	42.060118869	10.54.15.68	10.54.15.100	TCP	74	80 → 51012 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=4294948402 TSecr=2591273 WS=4
30	42.060140002	10.54.15.100	10.54.15.68	TCP	66	51012 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2591286 TSecr=4294948402
31	42.060253931	10.54.15.100	10.54.15.68	HTTP	501	POST /login.php HTTP/1.1 [application/x-www-form-urlencoded]
32	43.016754695	10.54.15.68	10.54.15.100	TCP	66	80 → 51012 [ACK] Seq=1 Ack=435 Win=15552 Len=0 TSval=4294949415 TSecr=2591286
33	43.033810817	10.54.15.68	10.54.15.100	HTTP	604	HTTP/1.1 200 OK (text/html)
34	43.033823724	10.54.15.100	10.54.15.68	TCP	66	51012 → 80 [ACK] Seq=436 Ack=539 Win=38336 Len=0 TSval=2591302 TSecr=4294949420
35	48.048277295	10.54.15.68	10.54.15.100	TCP	66	80 → 51012 [FIN, ACK] Seq=539 Ack=436 Win=15552 Len=0 TSval=4294950671 TSecr=2591302
36	48.048506072	10.54.15.100	10.54.15.68	TCP	66	51012 → 80 [FIN, ACK] Seq=436 Ack=540 Win=38336 Len=0 TSval=2592954 TSecr=4294950671
37	48.088082920	10.54.15.68	10.54.15.100	TCP	66	80 → 51012 [ACK] Seq=540 Ack=437 Win=15552 Len=0 TSval=4294950683 TSecr=2592954
38	91.903626490	fe80::2411:59ff:fe8...ff02::2		ICMPv6	70	Router Solicitation from 26:11:59:88:53:02

OUTLINE

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

- Another way this can be done is by using a display filter and increment the number of the TCP Stream. You will identify a non-valid TCP Stream by the non-visible packets.

No.	Time	Source	Destination	Protocol	Length	Info
5	0.000000000	10.54.15.66	10.54.15.66	TCP	74	80 → 50982 [SYN, ACK] Seq=1
6	0.000011102	10.54.15.100	10.54.15.66	TCP	66	50982 → 80 [ACK] Seq=1
7	0.000014128	10.54.15.100	10.54.15.66	HTTP	347	GET / HTTP/1.1
8	0.140951934	10.54.15.66	10.54.15.100	TCP	66	80 → 50982 [ACK] Seq=1
9	0.180990372	10.54.15.66	10.54.15.100	HTTP	654	HTTP/1.1 200 OK (text/
10	0.181016327	10.54.15.100	10.54.15.66	TCP	66	50982 → 80 [ACK] Seq=20
11	0.212109429	10.54.15.100	10.54.15.66	HTTP	345	GET /css/style.css HTTP
12	0.298521761	10.54.15.66	10.54.15.100	HTTP	1242	HTTP/1.1 200 OK (text/
13	0.327395188	10.54.15.100	10.54.15.66	TCP	66	50982 → 80 [ACK] Seq=50
14	5.290126996	10.54.15.66	10.54.15.100	TCP	66	80 → 50982 [FIN, ACK] S
15	5.290130000	10.54.15.100	10.54.15.66	TCP	66	50982 → 80 [FIN, ACK] S
16	5.330980407	10.54.15.66	10.54.15.100	TCP	66	80 → 50982 [ACK] Seq=1

No.	Time	Source	Destination	Protocol	Length	Info
26	42.917398520	10.54.15.100	10.54.15.68	TCP	74	51012 → 80 [SYN, Seq=
29	42.969116869	10.54.15.68	10.54.15.100	TCP	74	80 → 51012 [SYN, ACK=
30	42.969140002	10.54.15.100	10.54.15.68	TCP	66	51012 → 80 [ACK] Seq=
31	42.969353931	10.54.15.100	10.54.15.68	HTTP	501	POST /login.php HTTP/
32	43.016754695	10.54.15.68	10.54.15.100	TCP	66	80 → 51012 [ACK] Seq=
33	43.033810817	10.54.15.68	10.54.15.100	HTTP	604	HTTP/1.1 200 OK (text
34	43.033823724	10.54.15.100	10.54.15.68	TCP	66	51012 → 80 [ACK] Seq=
35	48.046277295	10.54.15.68	10.54.15.100	TCP	66	80 → 51012 [FIN, ACK=
36	48.040596072	10.54.15.100	10.54.15.68	TCP	66	51012 → 80 [FIN, ACK=
37	48.088082920	10.54.15.68	10.54.15.100	TCP	66	80 → 51012 [ACK] Seq=

No.	Time	Source	Destination	Protocol	Length	Info
19	26.862042221	10.54.15.100	10.54.15.68	TCP	14	51008 → 80 [RST] Seq=66
20	26.907980258	10.54.15.68	10.54.15.100	TCP	74	80 → 51008 [SYN, ACK]
21	26.967900940	10.54.15.100	10.54.15.68	TCP	60	51008 → 80 [ACK] Seq=1
21	26.989172432	10.54.15.100	10.54.15.68	HTTP	358	GET /favicon.ico HTTP/1.1
22	26.955928739	10.54.15.68	10.54.15.100	TCP	60	80 → 51008 [ACK] Seq=1
23	26.957862825	10.54.15.68	10.54.15.100	HTTP	568	HTTP/1.1 404 Not Found
24	26.957877026	10.54.15.100	10.54.15.68	TCP	66	51008 → 80 [ACK] Seq=2
25	26.962882070	10.54.15.68	10.54.15.100	TCP	60	80 → 51008 [FIN, ACK]
26	26.962789254	10.54.15.100	10.54.15.68	TCP	66	51008 → 80 [FIN, ACK]
27	26.008812475	10.54.15.68	10.54.15.100	TCP	66	80 → 51008 [ACK] Seq=5

No.	Time	Source	Destination	Protocol	Length	Info
tcp.stream eq 3						

OUTLINE

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

3. Under the TCP tab in Conversations we can see there are 3 TCP Streams. From here we can select a stream and choose Follow Stream from bottom right corner.

Ethernet · 4		IPv4 · 1		IPv6 · 1		TCP · 3		UDP					
Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.54.15.100	50982	10.54.15.68	80	13	3198	7	1030	6	2168	0.050863	5.2850	1559	
10.54.15.100	51008	10.54.15.68	80	10	1470	5	630	5	840	20.862049	5.1468	979	
10.54.15.100	51012	10.54.15.68	80	10	1649	5	773	5	876	42.917399	5.1707	1195	



OUTLINE

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

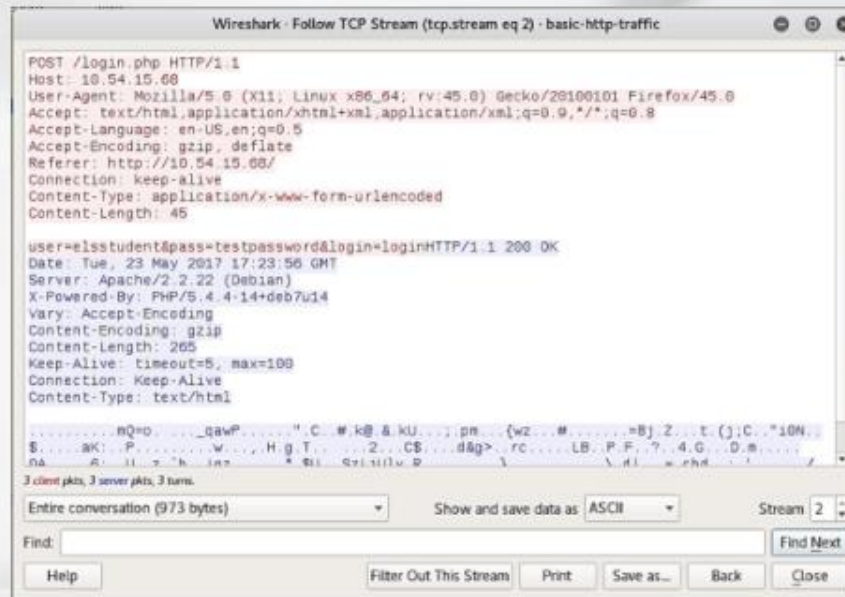
2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

- At the bottom right we can see a box, there we can select the TCP stream we would like to view.

By inspecting each of the TCP Streams we can see from the image that the 3rd TCP Stream (zero indexed) will contain the cleartext credentials. This is indeed valid HTTP traffic and typical HTTP behavior.



OUTLINE

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

▼ 2.2.2.1 HTTP

2.2.2.1.1 Malicious HTTP Traffic Example

Now we'll look at an example of suspicious traffic via port 80 using Wireshark and the techniques already demonstrated.

Remember the tips regarding normal HTTP traffic:

- Typically port 80.
- Cleartext web-based traffic.
- Hosts are accessed using FQDNs instead of IP addresses.

Shared example with the [THP](#) course

OUTLINE

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

▼ 2.2.2.1 HTTP

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Example

Detection

Before we dive deeper, let's take a moment to give the traffic capture below a look and see if we can spot anything that might look odd.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.124.211.200	10.124.211.96	TCP	74	33020 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=125613 TSecr=0 WS=128
2	0.056720	10.124.211.96	10.124.211.200	TCP	74	80 → 33020 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=222206 TSecr=125613 WS=4
3	0.056747	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=125627 TSecr=222206
4	0.056824	10.124.211.200	10.124.211.96	HTTP	349	GET / HTTP/1.1
5	0.133531	10.124.211.96	10.124.211.200	TCP	66	80 → 33020 [ACK] Seq=1 Ack=204 Win=15552 Len=0 TSval=222223 TSecr=125627
6	0.133549	10.124.211.96	10.124.211.200	HTTP	101	[TCP Previous segment not captured] Continuation
7	0.133556	10.124.211.200	10.124.211.96	TCP	78	[TCP Window Update] 33020 → 80 [ACK] Seq=284 Ack=1 Win=30336 Len=0 TSval=125647 TSecr=222223 SLE=1320 SRE=1361
8	0.150200	10.124.211.96	10.124.211.200	TCP	1391	[TCP Out-of-order] 80 → 33020 [ACK] Seq=1 Ack=204 Win=15552 Len=1320 TSval=222224 TSecr=125627
9	0.150259	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [ACK] Seq=284 Ack=1361 Win=33280 Len=0 TSval=125652 TSecr=222224
10	3.481495	10.124.211.200	10.124.211.96	HTTP	389	GET /news.php HTTP/1.1
11	3.582272	10.124.211.96	10.124.211.200	TCP	1391	[TCP segment of a reassembled PDU]
12	3.582305	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [ACK] Seq=607 Ack=2686 Win=36096 Len=0 TSval=126501 TSecr=223083
13	3.582578	10.124.211.96	10.124.211.200	HTTP	1953	HTTP/1.1 200 OK (text/html)
14	3.582589	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [ACK] Seq=607 Ack=3973 Win=39040 Len=0 TSval=126501 TSecr=223083
15	5.968993	10.124.211.200	10.124.211.96	HTTP	410	GET /newsdetails.php?id=26 HTTP/1.1
16	6.020439	10.124.211.96	10.124.211.200	TCP	1391	[TCP segment of a reassembled PDU]
17	6.020461	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [ACK] Seq=951 Ack=5298 Win=41856 Len=0 TSval=127118 TSecr=223701
18	6.020470	10.124.211.96	10.124.211.200	HTTP	83	HTTP/1.1 200 OK (text/html)
19	6.020471	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [ACK] Seq=951 Ack=5315 Win=41856 Len=0 TSval=127118 TSecr=223701
20	9.453656	10.124.211.200	10.124.211.96	HTTP	373	GET /newsdetails.php?id=26&27 HTTP/1.1
21	9.517192	10.124.211.96	10.124.211.200	TCP	1391	[TCP segment of a reassembled PDU]
22	9.517210	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [ACK] Seq=1258 Ack=6640 Win=44800 Len=0 TSval=127992 TSecr=224575
23	9.517216	10.124.211.96	10.124.211.200	HTTP	89	HTTP/1.1 200 OK (text/html)
24	9.517218	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [ACK] Seq=1258 Ack=6642 Win=44800 Len=0 TSval=127992 TSecr=224575
25	14.520484	10.124.211.96	10.124.211.200	TCP	66	80 → 33020 [FIN, ACK] Seq=6642 Ack=1258 Win=18768 Len=0 TSval=225826 TSecr=127992
26	14.520662	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [FIN, ACK] Seq=1258 Ack=6643 Win=44800 Len=0 TSval=129243 TSecr=225826
27	14.582082	10.124.211.96	10.124.211.200	TCP	66	80 → 33020 [ACK] Seq=6643 Ack=1259 Win=18768 Len=0 TSval=225842 TSecr=129243
28	17.437742	10.124.211.200	10.124.211.96	TCP	74	33022 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=129973 TSecr=0 WS=128
29	17.583661	10.124.211.96	10.124.211.200	TCP	74	80 → 33022 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=129572 TSecr=129973 WS=4

Shared example with the [THP](#) course

OUTLINE

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

▼ 2.2.2.1 HTTP

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Example

Detection

Shared example with the [THP](#) course

There is one thing that stands out right away which is highlighted below.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.124.211.200	10.124.211.96	TCP	74	33020 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=125613 TSecr=0 WS=128
2	0.056720	10.124.211.96	10.124.211.200	TCP	74	80 → 33020 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=222206 TSecr=125613 WS=4
3	0.056747	10.124.211.200	10.124.211.96	TCP	60	33020 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=125627 TSecr=222206
4	0.056824	10.124.211.200	10.124.211.96	HTTP	349	GET / HTTP/1.1
5	0.133531	10.124.211.96	10.124.211.200	TCP	60	80 → 33020 [ACK] Seq=1 Ack=284 Win=15552 Len=0 TSval=222223 TSecr=125627
6	0.133549	10.124.211.96	10.124.211.200	HTTP	101	[TCP Previous segment not captured] Continuation
7	0.133556	10.124.211.200	10.124.211.96	TCP	78	[TCP window update] 33020 → 80 [ACK] Seq=284 Ack=1 Win=36336 Len=0 TSval=125647 TSecr=222223 SIF=1326 SRF=1361
8	0.156230	10.124.211.96	10.124.211.200	TCP	1301	[TCP Out-Of-Order] 80 → 33020 [ACK] Seq=1 Ack=284 Win=15552 Len=1325 TSval=125627 TSecr=125627
9	0.156250	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [ACK] Seq=284 Ack=1361 Win=33280 Len=0 TSval=125652 TSecr=222224
10	3.481485	10.124.211.200	10.124.211.96	HTTP	389	GET /news.php HTTP/1.1
11	3.552272	10.124.211.96	10.124.211.200	TCP	1391	[TCP segment of a reassembled PDU]
12	3.552305	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [ACK] Seq=607 Ack=2686 Win=36096 Len=0 TSval=126501 TSecr=223083
13	3.552578	10.124.211.96	10.124.211.200	HTTP	1353	HTTP/1.1 200 OK (text/html)
14	3.552589	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [ACK] Seq=607 Ack=3973 Win=39040 Len=0 TSval=126501 TSecr=223083
15	5.068993	10.124.211.200	10.124.211.96	HTTP	410	GET /newsdetails.php?id=26 HTTP/1.1
16	5.020439	10.124.211.96	10.124.211.200	TCP	1391	[TCP segment of a reassembled PDU]
17	5.020461	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [ACK] Seq=951 Ack=5298 Win=41856 Len=0 TSval=127118 TSecr=223701
18	5.020470	10.124.211.96	10.124.211.200	HTTP	83	HTTP/1.1 200 OK (text/html)
19	5.020471	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [ACK] Seq=951 Ack=5315 Win=41856 Len=0 TSval=127118 TSecr=223701
20	9.453050	10.124.211.200	10.124.211.96	HTTP	373	GET /newsdetails.php?id=26%27 HTTP/1.1
21	9.517192	10.124.211.96	10.124.211.200	TCP	1391	[TCP segment of a reassembled PDU]
22	9.517210	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [ACK] Seq=1258 Ack=6640 Win=44800 Len=0 TSval=127992 TSecr=224575
23	9.517216	10.124.211.96	10.124.211.200	HTTP	68	HTTP/1.1 200 OK (text/html)
24	9.517218	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [ACK] Seq=1258 Ack=6642 Win=44800 Len=0 TSval=127992 TSecr=224575
25	14.520484	10.124.211.96	10.124.211.200	TCP	66	80 → 33020 [FIN, ACK] Seq=6642 Ack=1258 Win=18768 Len=0 TSval=225826 TSecr=127992
26	14.520662	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [FIN, ACK] Seq=1258 Ack=6643 Win=44800 Len=0 TSval=129243 TSecr=225826
27	14.582082	10.124.211.96	10.124.211.200	TCP	66	80 → 33020 [ACK] Seq=6643 Ack=1259 Win=18768 Len=0 TSval=225842 TSecr=129243
28	17.437742	10.124.211.200	10.124.211.96	TCP	74	33022 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=129973 TSecr=0 WS=128
29	17.503661	10.124.211.96	10.124.211.200	TCP	74	80 → 33022 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=226572 TSecr=129973 WS=4

- In the screenshot we can quickly see normal requests to web pages such as news.php and newsdetails.php?id=26 then we see another request to **newsdetails.php?id=26%27**
- If we URL decode %27 it corresponds to a **single quote (')**. This behavior is typical when looking for a [SQL injection](#). Now was someone looking for a SQLi or did someone make a typo in their web page request? Let's go further down the capture file to find out.

<https://www.acunetix.com/websitesecurity/sql-injection/>

IHRPv1 - Caendra Inc. © 2018 | p.125

OUTLINE

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

▼ 2.2.2.1 HTTP

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Example

Detection

Glancing further at the capture file, a few packets down, we can see more indications of SQL injection attempts. Looking at packets #32 and #44 we can see that the person didn't make a typo request at packet #20. This is indeed an attempt to find SQL injection on the server.

No.	Time	Source	Destination	Protocol	Length	Info
20	9.453656	10.124.211.200	10.124.211.96	HTTP	373	GET /newsdetails.php?id=20&27 HTTP/1.1
21	9.517192	10.124.211.96	10.124.211.200	TCP	1391	[TCP segment of a reassembled PDU]
22	9.517210	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [ACK] Seq=1258 Ack=6640 Win=44800 Len=0 TSval=127992 TSecr=224575
23	9.517216	10.124.211.96	10.124.211.200	HTTP	66	HTTP/1.1 200 OK (text/html)
24	9.517218	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [ACK] Seq=1258 Ack=6642 Win=44800 Len=0 TSval=127992 TSecr=224575
25	14.520484	10.124.211.96	10.124.211.200	TCP	66	80 → 33020 [FIN, ACK] Seq=6642 Ack=1258 Win=18768 Len=0 TSval=225826 TSecr=127992
26	14.520662	10.124.211.200	10.124.211.96	TCP	66	33020 → 80 [FIN, ACK] Seq=1258 Ack=6643 Win=44800 Len=0 TSval=129243 TSecr=225826
27	14.582082	10.124.211.96	10.124.211.200	TCP	66	80 → 33020 [ACK] Seq=6643 Ack=1259 Win=18768 Len=0 TSval=225842 TSecr=129243
28	17.437742	10.124.211.200	10.124.211.96	TCP	74	33022 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=129073 TSecr=0 WS=128
29	17.503661	10.124.211.96	10.124.211.200	TCP	74	80 → 33022 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=226572 TSecr=129073 WS=4
30	17.503697	10.124.211.200	10.124.211.96	TCP	66	33022 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=129080 TSecr=226572
31	17.504112	10.124.211.200	10.124.211.96	HTTP	389	GET /newsdetails.php?id=20&20and%201=1;--%20 HTTP/1.1
32	17.575934	10.124.211.96	10.124.211.200	TCP	66	80 → 33022 [ACK] Seq=1 Ack=324 Win=15552 Len=0 TSval=226590 TSecr=129080
33	17.578773	10.124.211.96	10.124.211.200	TCP	84	[TCP Previous segment not captured] [TCP segment of a reassembled PDU]
34	17.578787	10.124.211.200	10.124.211.96	TCP	78	[TCP Window Update] 33022 → 80 [ACK] Seq=324 Ack=1 Win=30336 Len=0 TSval=130008 TSecr=226590 SLE=1326 SRE=1344
35	17.579058	10.124.211.96	10.124.211.200	TCP	1391	[TCP Out-Of-Order] 80 → 33022 [ACK] Seq=1 Ack=324 Win=15552 Len=1326 TSval=226591 TSecr=129080
36	17.579085	10.124.211.200	10.124.211.96	TCP	66	33022 → 80 [ACK] Seq=324 Ack=1344 Win=33280 Len=0 TSval=130008 TSecr=226591
37	22.579448	10.124.211.200	10.124.211.96	TCP	66	33022 → 80 [FIN, ACK] Seq=324 Ack=1344 Win=33280 Len=0 TSval=131258 TSecr=226591
38	22.584131	10.124.211.96	10.124.211.200	TCP	66	80 → 33022 [FIN, ACK] Seq=1344 Ack=324 Win=15552 Len=0 TSval=227842 TSecr=130008
39	22.584144	10.124.211.200	10.124.211.96	TCP	66	33022 → 80 [ACK] Seq=325 Ack=1345 Win=33280 Len=0 TSval=131259 TSecr=227842
40	22.629046	10.124.211.96	10.124.211.200	TCP	66	80 → 33022 [ACK] Seq=1345 Ack=325 Win=15552 Len=0 TSval=227852 TSecr=131258
41	25.101234	10.124.211.200	10.124.211.96	TCP	74	33024 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=131888 TSecr=0 WS=128
42	25.141132	10.124.211.96	10.124.211.200	TCP	74	80 → 33024 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=228461 TSecr=131888 WS=4
43	25.141157	10.124.211.200	10.124.211.96	TCP	66	33024 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=131896 TSecr=228461
44	25.141361	10.124.211.200	10.124.211.96	HTTP	389	GET /newsdetails.php?id=20&20and%201=1;--%20 HTTP/1.1
45	25.188826	10.124.211.96	10.124.211.200	TCP	66	80 → 33024 [ACK] Seq=1 Ack=324 Win=15552 Len=0 TSval=228492 TSecr=131899
46	25.187197	10.124.211.96	10.124.211.200	HTTP	1285	HTTP/1.1 200 OK (text/html)
47	25.187168	10.124.211.200	10.124.211.96	TCP	66	33024 → 80 [ACK] Seq=324 Ack=1220 Win=32128 Len=0 TSval=131910 TSecr=228493
48	30.188296	10.124.211.200	10.124.211.96	TCP	66	33024 → 80 [FIN, ACK] Seq=324 Ack=1220 Win=32128 Len=0 TSval=133160 TSecr=228493

Shared example with the [THP](#) course

OUTLINE

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

▼ 2.2.2.1 HTTP

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Example

Detection

Now knowing that the individual is conducting SQL injection queries against the server, is he/she doing this manually or by using a tool?

Shared example with the [THP](#) course

IHRPv1 - Caendra Inc. © 2018 | p.127

OUTLINE

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

▼ 2.2.2.1 HTTP

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Example

Detection

We begin at the first packet in question, packet #20. By further inspection, we see that the User-Agent is Firefox and the OS is Linux. User-Agent's can be spoofed, but not always, as we will see shortly.

```
▶ Frame 20: 373 bytes on wire (2984 bits), 373 bytes captured (2984 bits) on 0
▶ Ethernet II, Src: 1a:3a:46:bf:43:91 (1a:3a:46:bf:43:91), Dst: Vmware_a1:4e:f0 (00:50:56:a1:4e:f0)
▶ Internet Protocol Version 4, Src: 10.124.211.200, Dst: 10.124.211.96
▶ Transmission Control Protocol, Src Port: 33020, Dst Port: 80, Seq: 951, Ack: 5315, Len: 307
▶ Hypertext Transfer Protocol
  ▶ GET /newsdetails.php?id=26%27 HTTP/1.1\r\n
    Host: 10.124.211.96\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
    \r\n
    [Full request URI: http://10.124.211.96/newsdetails.php?id=26%27]
    [HTTP request 4/4]
    [Prev request in frame: 15]
    [Response in frame: 23]
```

Shared example with the THP course

OUTLINE

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

▼ 2.2.2.1 HTTP

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Example

Detection

We see that the User-Agent is the same for packets #31 and #44.

Let's keep looking to see if we can find anything else of interest.

```
▶ Frame 44: 389 bytes on wire (3112 bits), 389 bytes captured (3112 bits) on interface 0
▶ Ethernet II, Src: 1a:3a:46:bf:43:91 (1a:3a:46:bf:43:91), Dst: Vmware_a1:4e:f0 (00:50:56:a1:4e:f0)
▶ Internet Protocol Version 4, Src: 10.124.211.200, Dst: 10.124.211.96
▶ Transmission Control Protocol, Src Port: 33024, Dst Port: 80, Seq: 1, Ack: 1, Len: 323
▶ Hypertext Transfer Protocol
  ▶ GET /newsdetails.php?id=26%20and%201=2;--%20- HTTP/1.1\r\n
    Host: 10.124.211.96\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
    \r\n
    [Full request URI: http://10.124.211.96/newsdetails.php?id=26%20and%201=2;--%20-]
    [HTTP request 1/1]
    [Response in frame: 46]
```

```
▶ Frame 31: 389 bytes on wire (3112 bits), 389 bytes captured (3112 bits) on interface 0
▶ Ethernet II, Src: 1a:3a:46:bf:43:91 (1a:3a:46:bf:43:91), Dst: Vmware_a1:4e:f0 (00:50:56:a1:4e:f0)
▶ Internet Protocol Version 4, Src: 10.124.211.200, Dst: 10.124.211.96
▶ Transmission Control Protocol, Src Port: 33022, Dst Port: 80, Seq: 1, Ack: 1, Len: 323
▶ Hypertext Transfer Protocol
  ▶ GET /newsdetails.php?id=26%20and%201=1;--%20- HTTP/1.1\r\n
    Host: 10.124.211.96\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
    \r\n
    [Full request URI: http://10.124.211.96/newsdetails.php?id=26%20and%201=1;--%20-]
    [HTTP request 1/1]
```

Shared example with the [THP](#) course

OUTLINE

2.2.2.1 HTTP

2.2.2.1 HTTP

2.2.2.1 HTTP

▼ 2.2.2.1 HTTP

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Example

Detection

By looking back to the initial packet of interest, packet #20, we can look at the Line-based **text data: text/html** portion of the packet details and notice that error messages are being displayed to the individual conducting the SQL injection queries.

```
\t<!-- end sidebar -->\n\t<div id="content">\n\t\t<div></div>\n\t\t<div class="boxed">\n\t\t\t<h1 class="title2">\n\t\t\t\n\t\t\t<br />\n\t\t\t<b>Warning</b>: mysqli_fetch_array() expects parameter 1 to be mysqli_result, boolean given in <b>/var/www/newsdetails.php</b> on line <b>11</b><br />\n\t\t\t\n\t\t\t</h1>\n\t\t\t<p>\n\t\t\t<br />\n\t\t\t<b>Warning</b>: mysqli_fetch_array() expects parameter 1 to be mysqli_result, boolean given in <b>/var/www/newsdetails.php</b> on line <b>20</b><br />\n\t\t\t</p>\n\t\t\t\n\t\t</div>\n
```

Shared example with the [THP](#) course

OUTLINE

2.2.2.1 HTTP

2.2.2.1 HTTP

▼ 2.2.2.1 HTTP

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Example

Detection

Strangely, packet #56, and the packet after that, packet #73, don't seem to contain any SQL injection queries. Maybe the individual decided not to continue the attack?

No.	Time	Source	Destination	Protocol	Length	Info
56	34.163969	10.124.211.200	10.124.211.96	HTTP	266	GET /newsdetails.php?id=1 HTTP/1.1
57	34.169322	10.124.211.96	10.124.211.200	TCP	74	80 → 33028 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=230739 TSecr=134145 WS=4
58	34.169364	10.124.211.200	10.124.211.96	TCP	66	33028 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=134156 TSecr=230739
59	34.180457	10.124.211.200	10.124.211.96	TCP	74	33030 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=134158 TSecr=0 WS=128
60	34.202983	10.124.211.96	10.124.211.200	TCP	66	80 → 33026 [ACK] Seq=1 Ack=201 Win=15552 Len=0 TSval=230747 TSecr=134154
61	34.206414	10.124.211.96	10.124.211.200	TCP	1391	[TCP segment of a reassembled PDU]
62	34.206432	10.124.211.200	10.124.211.96	TCP	66	33026 → 80 [ACK] Seq=201 Ack=1326 Win=32128 Len=0 TSval=134165 TSecr=230748
63	34.206495	10.124.211.96	10.124.211.200	HTTP	82	HTTP/1.1 200 OK (text/html)
64	34.206501	10.124.211.200	10.124.211.96	TCP	66	33026 → 80 [ACK] Seq=201 Ack=1342 Win=32128 Len=0 TSval=134165 TSecr=230748
65	34.206508	10.124.211.96	10.124.211.200	TCP	66	80 → 33026 [FIN, ACK] Seq=1342 Ack=201 Win=15552 Len=0 TSval=230748 TSecr=134154
66	34.207131	10.124.211.200	10.124.211.96	TCP	66	33026 → 80 [FIN, ACK] Seq=201 Ack=1343 Win=32128 Len=0 TSval=134165 TSecr=230748
67	34.218945	10.124.211.96	10.124.211.200	TCP	74	80 → 33030 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=230751 TSecr=134158 WS=4
68	34.218972	10.124.211.200	10.124.211.96	TCP	66	33030 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=134165 TSecr=230751
69	34.230622	10.124.211.200	10.124.211.96	TCP	74	33032 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=134171 TSecr=0 WS=128
70	34.248217	10.124.211.96	10.124.211.200	TCP	66	80 → 33026 [ACK] Seq=1343 Ack=202 Win=15552 Len=0 TSval=230758 TSecr=134165
71	34.269905	10.124.211.96	10.124.211.200	TCP	74	80 → 33032 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=230764 TSecr=134171 WS=4
72	34.269934	10.124.211.200	10.124.211.96	TCP	66	33032 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=134181 TSecr=230764
73	35.126264	10.124.211.200	10.124.211.96	HTTP	266	GET /newsdetails.php?id=1 HTTP/1.1
74	35.126569	10.124.211.200	10.124.211.96	TCP	74	33034 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=134395 TSecr=0 WS=128
75	35.166226	10.124.211.96	10.124.211.200	TCP	66	80 → 33028 [ACK] Seq=1 Ack=201 Win=15552 Len=0 TSval=230988 TSecr=134395
76	35.166267	10.124.211.96	10.124.211.200	TCP	74	80 → 33034 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=230988 TSecr=134395 WS=4
77	35.166341	10.124.211.200	10.124.211.96	TCP	66	33034 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=134405 TSecr=230988
78	35.168806	10.124.211.96	10.124.211.200	TCP	1391	[TCP segment of a reassembled PDU]
79	35.168835	10.124.211.200	10.124.211.96	TCP	66	33028 → 80 [ACK] Seq=201 Ack=1326 Win=32128 Len=0 TSval=134405 TSecr=230988
80	35.168926	10.124.211.96	10.124.211.200	HTTP	82	HTTP/1.1 200 OK (text/html)
81	35.168943	10.124.211.200	10.124.211.96	TCP	66	33028 → 80 [ACK] Seq=201 Ack=1342 Win=32128 Len=0 TSval=134405 TSecr=230988
82	35.168962	10.124.211.96	10.124.211.200	TCP	66	80 → 33028 [FIN, ACK] Seq=1342 Ack=201 Win=15552 Len=0 TSval=230988 TSecr=134395
83	35.170154	10.124.211.200	10.124.211.96	TCP	66	33028 → 80 [FIN, ACK] Seq=201 Ack=1343 Win=32128 Len=0 TSval=134408 TSecr=230988

Shared example with the THP course

OUTLINE

2.2.2.1 HTTP

▼ 2.2.2.1 HTTP

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Example

Detection

Let's take a closer look at packet #56. Indeed no SQL injection attempt is visible within this packet, but something else should have already alarmed us, the User-Agent. The User-Agent for this HTTP GET Request is [Sqlmap](http://sqlmap.org). So the attacker didn't quit, he/she escalated.



```
▶ Frame 56: 266 bytes on wire (2128 bits), 266 bytes captured (2128 bits) on 0
▶ Ethernet II, Src: 1a:3a:46:bf:43:91 (1a:3a:46:bf:43:91), Dst: Vmware_a1:4e:f0 (00:50:56:a1:4e:f0)
▶ Internet Protocol Version 4, Src: 10.124.211.200, Dst: 10.124.211.96
▶ Transmission Control Protocol, Src Port: 3307, Dst Port: 80, Seq: 1, Ack: 1, Len: 200
▶ Hypertext Transfer Protocol
  ▶ GET /newsdetails.php?id=1 HTTP/1.1\r\n
    Accept-Encoding: gzip,deflate\r\n
    Host: 10.124.211.96\r\n
    Accept: */*\r\n
    User-Agent: sqlmap/1.1.4#stable (http://sqlmap.org)\r\n
    Connection: close\r\n
    Cache-Control: no-cache\r\n
  \r\n
  [Full request URI: http://10.124.211.96/newsdetails.php?id=1]
  [HTTP request 1/1]
  [Response in frame: 63]
```

Shared example with the [THP](#) course

OUTLINE

▼ 2.2.2.1 HTTP

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious
HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Example

Detection

A few of the subsequent HTTP GET Requests that follows will also look like legit HTTP traffic but these will also contain the same User-Agent within the packet details, Sqlmap.

In addition, they will most probably contain overly long URI requests.

Shared example with the [THP](#) course

IHRPv1 - Caendra Inc. © 2018 | p.133

OUTLINE

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.2 HTTPS

Now we'll look at HTTPS traffic within the upcoming slides.

We'll look into some pointers to remember about HTTPS traffic and some tips about normal and suspicious HTTPS traffic.

Finally we'll study traffic captures of both normal and suspicious HTTPS traffic.

OUTLINE

2.2.2.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

▼ 2.2.2.2 HTTPS

2.2.2.2 HTTPS

A few things to remember about HTTPS:

- HTTPS is the secure version of HTTP.
 - The “S” refers to Secure Socket Layer/Transport Layer Security (SSL/TLS).
- HTTPS also establishes a handshake similar to TCP but more complicated. Below is a brief summary:
 - Both the client and the server need to agree on the protocol version.
 - Both the client and the server need to select cryptographic algorithms.
 - Optionally authenticate to each other.
 - Use public key encryption techniques to establish secure communications.

OUTLINE

2.2.2.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

▼ 2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

Some facts to help distinguish normal and suspicious HTTPS traffic

Normal HTTPS Traffic	Suspicious HTTPS Traffic
Port 443, TCP Port 8443, TCP (used as alternate)	Malicious binaries (backdoors), scripts, web shells, etc. will use this port because typically in all corporate environments the port is open.
Encrypted traffic	If the traffic is not encrypted and Secure Sockets Layer packet details are empty within packet details then that will fall under suspicious.
Web server typically in FQDN format.	Server will point to an IP address instead of FQDN format.

OUTLINE

2.2.2.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

▼ 2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

We will not dive too deep into the handshake used by HTTPS traffic but there are certain details within the packets that we should be familiar with. Those details will be pointed out in the upcoming slides.

If you want to dive deeper into how HTTPS establishes secure communications please refer to [RFC 6101](https://tools.ietf.org/html/rfc6101) for SSL 3.0, [RFC 5246](https://tools.ietf.org/html/rfc5246) for TLS 1.2 and [RFC 8446](https://datatracker.ietf.org/doc/rfc8446/) for TLS 1.3.

<https://tools.ietf.org/html/rfc6101>
<https://tools.ietf.org/html/rfc5246>
<https://datatracker.ietf.org/doc/rfc8446/>

OUTLINE

2.2.2.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

▼ 2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

No.	Time	Source	Destination	Protocol	Length	Info
3	8.046179	10.54.15.100	10.54.15.15	TCP	74	39678 → 88 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2640968 TSecr=0 WS=128
9	14.560349	10.54.15.100	10.54.15.15	TCP	74	45112 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2644900 TSecr=0 WS=128
10	14.608349	10.54.15.100	10.54.15.15	TCP	74	443 → 45112 [SYN, ACK] Seq=0 Ack=1 Win=1460 Len=0 MSS=1337 SACK_PERM=1 TSval=2644600 TSecr=2644600 WS=4
11	14.608374	10.54.15.100	10.54.15.15	TCP	66	45112 → 443 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2644612 TSecr=35434
12	14.608618	10.54.15.100	10.54.15.15	TLSv1.2	233	Client Hello
13	14.654485	10.54.15.15	10.54.15.100	TCP	66	443 → 45112 [ACK] Seq=1 Ack=168 Win=15552 Len=0 TSval=35446 TSecr=2644612
14	14.658826	10.54.15.15	10.54.15.100	TLSv1.2	1391	Server Hello, Certificate
15	14.658839	10.54.15.100	10.54.15.15	TCP	66	45112 → 443 [ACK] Seq=168 Ack=1326 Win=32128 Len=0 TSval=2644624 TSecr=35447
16	14.660183	10.54.15.100	10.54.15.15	TLSv1.2	73	Alert (Level: Fatal, Description: Unknown CA)
17	14.662282	10.54.15.100	10.54.15.15	TCP	66	45112 → 443 [FIN, ACK] Seq=175 Ack=1326 Win=32128 Len=0 TSval=2644625 TSecr=35447
18	14.662516	10.54.15.15	10.54.15.100	TLSv1.2	170	Server Key Exchange, Server Hello Done
19	14.663522	10.54.15.100	10.54.15.15	TCP	66	45112 → 443 [RST] Seq=168 Win=0 Len=0
20	14.706639	10.54.15.15	10.54.15.100	TCP	66	443 → 45112 [FIN, ACK] Seq=1430 Ack=176 Win=15552 Len=0 TSval=35459 TSecr=2644625
21	14.706658	10.54.15.100	10.54.15.15	TCP	66	45112 → 443 [RST] Seq=176 Win=0 Len=0
22	22.545448	10.54.15.100	10.54.15.15	TCP	74	45114 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2646500 TSecr=0 WS=128
23	22.554263	10.54.15.15	10.54.15.100	TCP	74	443 → 45114 [SYN, ACK] Seq=0 Ack=1 Win=1460 Len=0 MSS=1337 SACK_PERM=1 TSval=2646506 TSecr=2646506 WS=4
24	22.584291	10.54.15.100	10.54.15.15	TCP	66	45114 → 443 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2646608 TSecr=37438
25	22.594467	10.54.15.100	10.54.15.15	TLSv1.2	233	Client Hello
26	22.643572	10.54.15.15	10.54.15.100	TCP	66	443 → 45114 [ACK] Seq=1 Ack=168 Win=15552 Len=0 TSval=37443 TSecr=2646608
27	22.646963	10.54.15.15	10.54.15.100	TLSv1.2	1391	Server Hello, Certificate
28	22.646975	10.54.15.100	10.54.15.15	TCP	66	45114 → 443 [ACK] Seq=168 Ack=1326 Win=32128 Len=0 TSval=2646621 TSecr=37445
29	22.649560	10.54.15.15	10.54.15.100	TLSv1.2	170	Server Key Exchange, Server Hello Done
30	22.649568	10.54.15.100	10.54.15.15	TCP	66	45114 → 443 [ACK] Seq=168 Ack=1480 Win=32128 Len=0 TSval=2646622 TSecr=37445
31	22.651191	10.54.15.100	10.54.15.15	TLSv1.2	192	Client Key Exchange, Change Cipher Spec, Hello Request, Hello Request
32	22.651238	10.54.15.100	10.54.15.15	TLSv1.2	376	Application Data
33	22.706218	10.54.15.15	10.54.15.100	TCP	66	443 → 45114 [ACK] Seq=1430 Ack=604 Win=16624 Len=0 TSval=37457 TSecr=2646622
34	22.706636	10.54.15.15	10.54.15.100	TLSv1.2	324	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
35	22.704803	10.54.15.15	10.54.15.100	TLSv1.2	770	Application Data, Application Data, Application Data
36	22.704970	10.54.15.100	10.54.15.15	TCP	66	45114 → 443 [ACK] Seq=604 Ack=2392 Win=37564 Len=0 TSval=2646636 TSecr=37457
37	22.715752	10.54.15.100	10.54.15.15	TLSv1.2	376	Application Data
38	22.763897	10.54.15.15	10.54.15.100	TLSv1.2	1358	Application Data, Application Data, Application Data, Application Data
39	22.807472	10.54.15.100	10.54.15.15	TCP	66	45114 → 443 [ACK] Seq=613 Ack=3684 Win=40320 Len=0 TSval=2646662 TSecr=37474
40	27.767987	10.54.15.100	10.54.15.15	TLSv1.2	87	Encrypted Alert
41	27.768027	10.54.15.100	10.54.15.15	TCP	66	45114 → 443 [FIN, ACK] Seq=644 Ack=3684 Win=40320 Len=0 TSval=2647902 TSecr=37474
42	27.771149	10.54.15.15	10.54.15.100	TLSv1.2	87	Encrypted Alert
43	27.771188	10.54.15.100	10.54.15.15	TCP	66	45114 → 443 [RST] Seq=613 Win=0 Len=0

- When analyzing HTTPS traffic, we should see encrypted packets. All traffic on port 443 should look this way.

OUTLINE

2.2.2.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

▼ 2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

Let's now take a glance at packets for HTTPS traffic.

OUTLINE

2.2.2.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

▼ 2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

As you may remember, the **Secure Sockets Layer** portion of the packet details should **not** be empty. Here we see pertinent info as to what the client has available in order to attempt to establish secure communications with the server, **Client Hello packet**. We see the following:

- Content Type = Handshake
- Handshake Protocol: Client Hello
- Version: TLS 1.2
- Cipher Suites (11 suites)
- Compression Method (1 method)

```
▶ Frame 25: 233 bytes on wire (1864 bits), 233 bytes captured (1864 bits) on 0
▶ Ethernet II, Src: 26:11:59:BB:53:02 (26:11:59:BB:53:02), Dst: Vmware_a1:61:66 (00:50:56:a1:61:66)
▶ Internet Protocol Version 4, Src: 10.54.15.100, Dst: 10.54.15.15
▶ Transmission Control Protocol, Src Port: 45114, Dst Port: 443, Seq: 1, Ack: 1, Len: 167
▼ Secure Sockets Layer
  ▼ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 162
    ▼ Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 158
      Version: TLS 1.2 (0x0303)
      ▶ Random
      Session ID Length: 0
      Cipher Suites Length: 22
      ▼ Cipher Suites (11 suites)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
        Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
        Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
        Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
        Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
        Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
      ▶ Compression Methods Length: 1
      ▼ Compression Methods (1 method)
        Compression Method: null (0)
      Extensions Length: 95
      ▶ Extension: renegotiation_info
      ▶ Extension: elliptic_curves
      ▶ Extension: ec_point_formats
      ▶ Extension: SessionTicket_TLS
      ▶ Extension: next_protocol_negotiation
      ▶ Extension: Application Layer Protocol Negotiation
      ▶ Extension: status_request
      ▶ Extension: signature_algorithms
```

OUTLINE

2.2.2.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

▼ 2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

Here is the server's response, **Server Hello** packet. These Hello messages again will contain the availability and agreement of encryption algorithms to work with and exchange of random values that will be used for generation of key.

```
Frame 27: 1391 bytes on wire (11128 bits), 1391 bytes captured (11128 bits) on 0
Ethernet II, Src: Vmware_a1:f4:d0 (00:50:56:a1:f4:d0), Dst: 26:11:59:08:53:02 (26:11:59:08:53:02)
Internet Protocol Version 4, Src: 10.54.15.15, Dst: 10.54.15.100
Transmission Control Protocol, Src Port: 443, Dst Port: 45114, Seq: 1, Ack: 169, Len: 1325
Secure Sockets Layer
  TLSv1.2 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 61
    Handshake Protocol: Server Hello
      Handshake Type: Server Hello (2)
      Length: 57
      Version: TLS 1.2 (0x0303)
      Random
        GMT Unix Time: May 23, 2017 19:27:38.000000000 EDT
        Random Bytes: 2080d7125ade0022e9441d5121c77b5e3cb08e05fd2242e...
      Session ID Length: 0
      Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
      Compression Method: null (0)
      Extensions Length: 17
      Extension: renegotiation_info
      Extension: ec_point_formats
      Extension: SessionTicket TLS
  TLSv1.2 Record Layer: Handshake Protocol: Certificate
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 1011
    Handshake Protocol: Certificate
      Handshake Type: Certificate (11)
      Length: 1007
      Certificates Length: 1004
      Certificates (1004 bytes)
        Certificate Length: 1001
        Certificate: 308203e5308202cda003020102020908d08303cf07501375... (pkcs-9-at-emailAddress=e
          signedCertificate
            version: v3 (2)
            serialNumber: -277336875500607947
            signature (sha1WithRSAEncryption)
            issuer: rdnSequence (0)
            validity
            subject: rdnSequence (0)
            subjectPublicKeyInfo
            extensions: 3 items
            algorithmIdentifier (sha1WithRSAEncryption)
            Padding: 0
            encrypted: 2326c0130a0c0a09ff904ee8e6909cae6f34ae00cf343ae9...
```

OUTLINE

2.2.2.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

▼ 2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

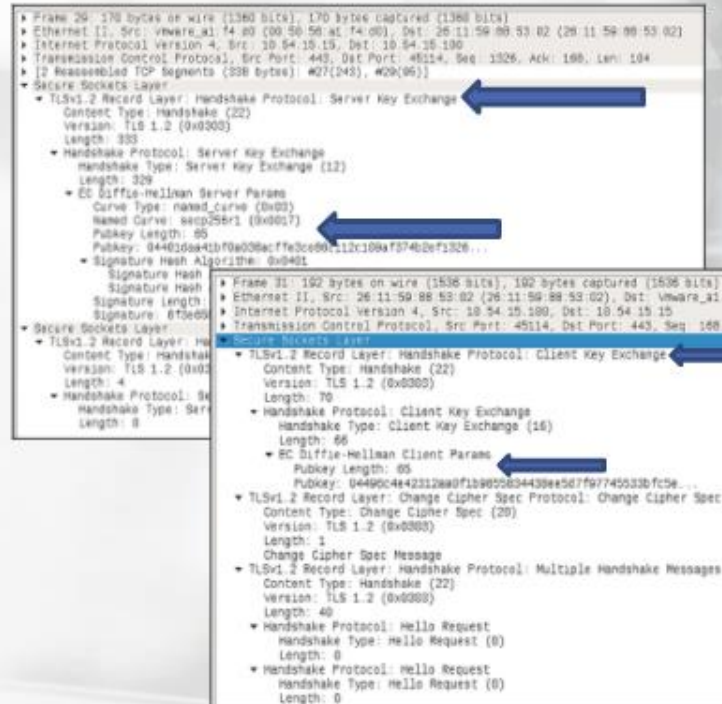
2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

Here we see the **Server Key Exchange** which will be followed by the **Client Key Exchange** packet.

This will be considered step #3 in the establishment of an SSL/TLS session between a server and a client.



OUTLINE

2.2.2.1.1 Malicious HTTP Traffic Exa...

2.2.2.1.1 Malicious HTTP Traffic Exa...

▼ 2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

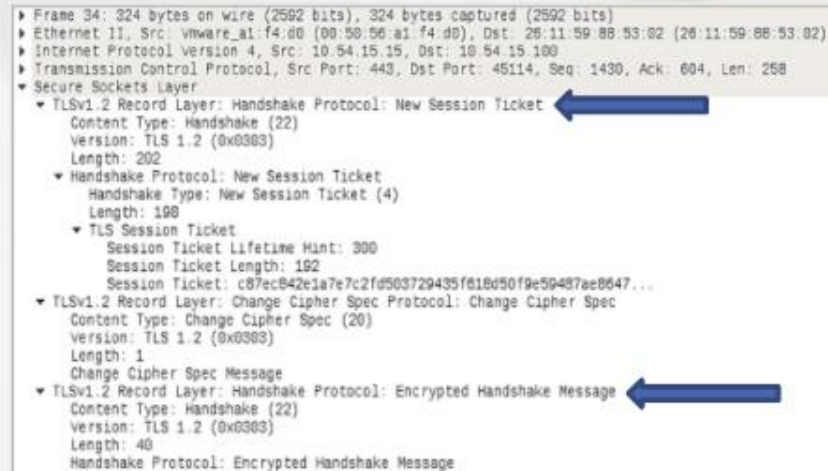
2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

This is the last packet and the handshake between the server and client is now complete.

The rest of the packets between these two devices will now be encrypted.



```
Frame 34: 324 bytes on wire (2592 bits), 324 bytes captured (2592 bits) on interface 0  
Ethernet II, Src: vmware_a1:f4:d0 (00:50:56:a1:f4:d0), Dst: 26:11:59:88:53:02 (26:11:59:88:53:02)  
Internet Protocol Version 4, Src: 10.54.15.15, Dst: 10.54.15.100  
Transmission Control Protocol, Src Port: 443, Dst Port: 45114, Seq: 1430, Ack: 604, Len: 258  
Secure Sockets Layer  
  TLSv1.2 Record Layer: Handshake Protocol: New Session Ticket ←  
    Content Type: Handshake (22)  
    Version: TLS 1.2 (0x0303)  
    Length: 202  
      Handshake Protocol: New Session Ticket  
        Handshake Type: New Session Ticket (4)  
        Length: 198  
          TLS Session Ticket  
            Session Ticket Lifetime Hint: 300  
            Session Ticket Length: 192  
            Session Ticket: c87ec842e1a7e7c2fd503729435f618d50f9e59487ae0647...  
  TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec  
    Content Type: Change Cipher Spec (20)  
    Version: TLS 1.2 (0x0303)  
    Length: 1  
    Change Cipher Spec Message  
  TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message ←  
    Content Type: Handshake (22)  
    Version: TLS 1.2 (0x0303)  
    Length: 40  
    Handshake Protocol: Encrypted Handshake Message
```

OUTLINE

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

▼ 2.2.2.2 HTTPS

2.2.2.2.1 Malicious
HTTPS Traffic Ex...

2.2.2.2 HTTPS

The traffic is unreadable, but if this is internal traffic within our corporate environment, then, it is feasible to decrypt this traffic using the private key from the internal server.

```
▶ Frame 35: 770 bytes on wire (6160 bits), 770 bytes captured (6160 bits)
▶ Ethernet II, Src: Vmware_a1:f4:d0 (00:50:56:a1:f4:d0), Dst: 26:11:59:88:53:02 (26:11:59:88:53:02)
▶ Internet Protocol Version 4, Src: 10.54.15.15, Dst: 10.54.15.100
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 45114, Seq: 1688, Ack: 604, Len: 704
▼ Secure Sockets Layer
  ▶ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
  ▶ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
  ▶ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
  ▼ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
    Content Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 32
    Encrypted Application Data: bb006752c8e53aef6bb13c15ff590c829883685da8b96e44...
```



OUTLINE

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

▼ 2.2.2.2 HTTPS

2.2.2.2.1 Malicious
HTTPS Traffic Ex...

2.2.2.2.1 Malicious HTTPS Traffic Example

Consider the *SSL_renegotiation.pcap* file found in this module's resources. Can you spot any abnormalities?

Hint: Apply the following filter in Wireshark.
`ssl.record.content_type == 22`

OUTLINE

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

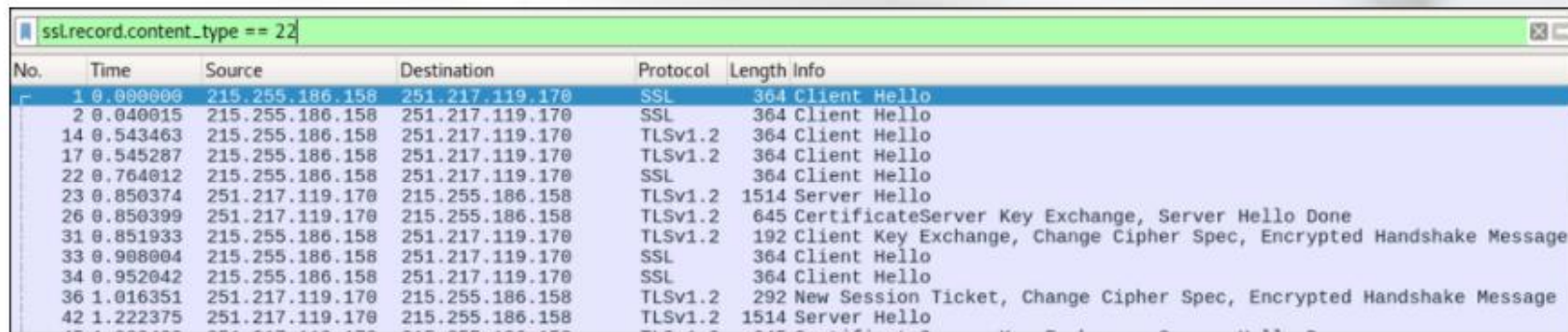
2.2.2.2 HTTPS

▼ 2.2.2.2 HTTPS

2.2.2.2.1 Malicious
HTTPS Traffic Ex...

2.2.2.2 HTTPS

Detection



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	215.255.186.158	251.217.119.170	SSL	364	Client Hello
2	0.040915	215.255.186.158	251.217.119.170	SSL	364	Client Hello
14	0.543463	215.255.186.158	251.217.119.170	TLSv1.2	364	Client Hello
17	0.545287	215.255.186.158	251.217.119.170	TLSv1.2	364	Client Hello
22	0.764012	215.255.186.158	251.217.119.170	SSL	364	Client Hello
23	0.850374	251.217.119.170	215.255.186.158	TLSv1.2	1514	Server Hello
26	0.850399	251.217.119.170	215.255.186.158	TLSv1.2	645	CertificateServer Key Exchange, Server Hello Done
31	0.851933	215.255.186.158	251.217.119.170	TLSv1.2	192	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
33	0.908004	215.255.186.158	251.217.119.170	SSL	364	Client Hello
34	0.952042	215.255.186.158	251.217.119.170	SSL	364	Client Hello
36	1.016351	251.217.119.170	215.255.186.158	TLSv1.2	292	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
42	1.222375	251.217.119.170	215.255.186.158	TLSv1.2	1514	Server Hello

- When it comes to SSL/TLS handshakes, you should remember two things:
 - Each SSL/TLS handshake is effectively a new connection (consuming resources)
 - SSL/TLS handshakes are quite CPU intensive operations (server-side)
- The number of new **Client Hello** messages is abnormal.
- Taking into consideration both the above, it looks like we are dealing with a [SSL Renegotiation Attack](#)

OUTLINE

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

▼ 2.2.2.2 HTTPS

2.2.2.2.1 Malicious
HTTPS Traffic Ex...

2.2.2.2 HTTPS

2.2.3 SMTP

SMTP (Simple Mail Transfer Protocol) is the internet's Postman.

It's the protocol responsible for sending emails.

OUTLINE

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

▼ 2.2.2.2 HTTPS

2.2.2.2.1 Malicious
HTTPS Traffic Ex...

2.2.2.2 HTTPS

▼ 2.2.3 SMTP

2.2.3 SMTP

When connecting to the SMTP server, the client has to identify itself and authenticate through a password.

Before discussing SMTP, it is important to remember that the protocol's implementation may differ from one product to another.

OUTLINE

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

▼ 2.2.2.2 HTTPS

2.2.2.2.1 Malicious
HTTPS Traffic Ex...

2.2.2.2 HTTPS

▼ 2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

SMTP is a text-based protocol, meaning that it relies on exchanging ASCII based strings as commands between the server and the client.

HELO	Sent by a client to identify itself, usually with a domain name.
MAIL FROM	Identifies the sender of the message; used in the form MAIL FROM:..
RCPT TO	Identifies the message recipients; used in the form RCPT TO:..
VRFY	Verifies that a mailbox is available for message delivery;.

OUTLINE

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

▼ 2.2.2.2 HTTPS

2.2.2.2.1 Malicious
HTTPS Traffic Ex...

2.2.2.2 HTTPS

▼ 2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

Later, in the 1995, ESMTP was developed. ESMTP was an extension to the original SMTP protocol with some modifications. Some of the modifications included new commands such as using EHLO instead of HELO for identification.

On your right, you can see an example of SMTP traffic.

```
Stream Content
220 WIN-DRAK19752C.DigitalForensics ESMTP Mailenable Service, version: 9.75-- ready at 08/28/17 19:40:48
EHLO [192.168.153.130]
250-DigitalForensics [192.168.153.130], this server offers 4 extensions
250-AUTH LOGIN
250-SIZE 40960000
250-HELP
250 AUTH=LOGIN
AUTH LOGIN
334 VxN1c2V5bW9udG9zaW50
dXN1c2V5bW9udG9zaW50
334 VxN1c2V5bW9udG9zaW50
VxN1c2V5bW9udG9zaW50
235 Authenticated
MAIL FROM:<user2@digitalforensics.els> SIZE=431
230 Requested mail action okay, completed
RCPT TO:<user2@digitalforensics.els>
230 Requested mail action okay, completed
DATA
254 Start mail input; end with <CRLF>.<CRLF>
To: user2@digitalforensics.els
From: user2 <user2@digitalforensics.els>
Subject: test
Message-ID: <d63ba2f2-eb4b-92cd-8bb4-44b213e9e63e@digitalforensics.els>
Date: Mon, 28 Aug 2017 19:40:47 +0300
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:52.0) Gecko/20100101
Thunderbird/52.3.0
X-MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8; format=flowed
Content-Transfer-Encoding: 7bit
Content-Language: en-US

test

230 Requested mail action okay, completed
QUIT
221 Service closing transmission channel

(Entire conversation (1130 bytes))
```

OUTLINE

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

▼ 2.2.2.2 HTTPS

2.2.2.2.1 Malicious
HTTPS Traffic Ex...

2.2.2.2 HTTPS

▼ 2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

The SMTP server starts the conversation, once the TCP three-way handshake is completed, by sending its banner, containing the server's name and version.

The above will look similar to the below on the wire.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000			TCP	62	1077→25 [SYN] Seq=0 Win=16384 Len=0 MSS=
2	0.000000			TCP	62	25→1077 [SYN, ACK] Seq=0 Ack=1 Win=175
3	0.020029			TCP	60	1077→25 [ACK] Seq=1 Ack=1 Win=17520 Len=
4	0.020029			SMTP	158	S: 220
5	0.030043			SMTP	67	C: EHLO Client
6	0.190274			TCP	54	25→1077 [ACK] Seq=105 Ack=14 Win=17507
7	0.420605			SMTP	290	S: 250 Server Hello [] 25
8	0.430619			SMTP	66	C: AUTH LOGIN
9	0.430619			SMTP	72	S: 334 VXNlcw5hbWw6
10	0.430619			SMTP	64	C: User: []
11	0.430619			SMTP	72	S: 334 UGFzc3dvcm06
12	0.430619			SMTP	64	C: Pass: []
13	0.440634			SMTP	91	S: 235 2.7.0 Authentication successful

OUTLINE

2.2.2.2 HTTPS

2.2.2.2 HTTPS

2.2.2.2 HTTPS

▼ 2.2.2.2 HTTPS

2.2.2.2.1 Malicious
HTTPS Traffic Ex...

2.2.2.2 HTTPS

▼ 2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

Back to our [example](#), the client starts the conversation with the **EHLO** command, which is the extended version of the **HELO** command.

The server replies with the options that an SMTP client can pick from.

Among these options is the authentication login option, where the client asks for authentication to login.

OUTLINE

2.2.2.2 HTTPS

2.2.2.2 HTTPS

▼ 2.2.2.2 HTTPS

2.2.2.2.1 Malicious
HTTPS Traffic Ex...

2.2.2.2 HTTPS

▼ 2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

Next, we can see the server asking the client for the username. However, the word *username* was encoded using base 64 encoding.

We can verify that by decoding the string which the server sent to the client.

Decode from Base64 format
Simply use the form below

VXNlcm5hbWU6

< DECODE > UTF-8 You may also select input charset.

☐ Live mode OFF Decodes while you type or paste (in strict mode).

Decodes an entire file (max. 10MB).

Username:

OUTLINE

2.2.2.2 HTTPS

▼ 2.2.2.2 HTTPS

2.2.2.2.1 Malicious
HTTPS Traffic Ex...

2.2.2.2 HTTPS

▼ 2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

The client responded to that request by sending their SMTP username encoded using base 64 encoding.

Decode from Base64 format

Simply use the form below

dXNlcjJAZGlnaXRhbGZvcnVuc2lscy5lbHM=

< DECODE >

UTF-8

You may also select input charset.

Live mode OFF

Decodes while you type or paste (in strict mode).

UPLOAD FILE

Decodes an entire file (max. 10MB).

user2@digitalforensics.els

OUTLINE

▼ 2.2.2.2 HTTPS

2.2.2.2.1 Malicious
HTTPS Traffic Ex...

2.2.2.2 HTTPS

▼ 2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

After receiving the username, the server asks the client for the password also in base 64 encoding.

Decode from Base64 format

Simply use the form below

UGFzc3dvcnQ6

< **DECODE** >

UTF-8

You may also select input charset.

☐ Live mode OFF

Decodes while you type or paste (in strict mode).

Decodes an entire file (max. 10MB).

Password:

OUTLINE

2.2.2.2.1 Malicious
HTTPS Traffic Ex...

2.2.2.2 HTTPS

▼ 2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

The client responded to that request by sending the password encoded in Base 64.

Decode from Base64 format

Simply use the form below

NDMyMQ==

< DECODE >

UTF-8

You may also select input charset.

☐ Live mode OFF

Decodes while you type or paste (in strict mode).

Decodes an entire file (max. 10MB).

4321

OUTLINE

2.2.2.1 SMTP

2.2.2.2 HTTPS

▼ 2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

Once both the username and the password is verified, the server acknowledges that, by telling the client that they have been "authenticated."

OUTLINE

▼ 2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

Using the **MAIL FROM** and **RCPT TO** command, the client tells the server that it wants to send an email.

After that, the mail body is specified using the **DATA** command.

OUTLINE

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

We can see that the email header specifies the **source**, **recipient**, **subject** and a **message ID**.

It also specifies the **date** and, more importantly, the **user-agent** field, which tells what mail client was used to send that email.

Finally, the type of content being sent is specified.

OUTLINE

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

It is important to mention that without the proper security configuration, an attacker may be able to manually connect to the SMTP server and send a forged email to a client either on the local SMTP server or to another SMTP server.

In such a case the SMTP server will act as an open relay.

OUTLINE

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

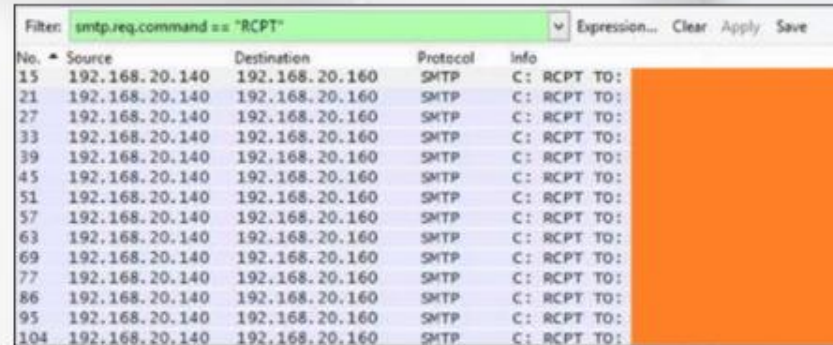
▼ 2.2.3 SMTP

2.2.3.1 Malicious SMTP Traffic Example

Detection

The lack of proper security configuration may also allow the attacker to connect to the SMTP server and manually enumerate the users on that server using the **VRIFY**, **EXPN** or **RCPT TO** commands.

User enumeration may be used as part of a social engineering attack or as a first step of a brute force attack against account passwords on that server.



No.	Source	Destination	Protocol	Info
15	192.168.20.140	192.168.20.160	SMTP	C: RCPT TO:
21	192.168.20.140	192.168.20.160	SMTP	C: RCPT TO:
27	192.168.20.140	192.168.20.160	SMTP	C: RCPT TO:
33	192.168.20.140	192.168.20.160	SMTP	C: RCPT TO:
39	192.168.20.140	192.168.20.160	SMTP	C: RCPT TO:
45	192.168.20.140	192.168.20.160	SMTP	C: RCPT TO:
51	192.168.20.140	192.168.20.160	SMTP	C: RCPT TO:
57	192.168.20.140	192.168.20.160	SMTP	C: RCPT TO:
63	192.168.20.140	192.168.20.160	SMTP	C: RCPT TO:
69	192.168.20.140	192.168.20.160	SMTP	C: RCPT TO:
77	192.168.20.140	192.168.20.160	SMTP	C: RCPT TO:
86	192.168.20.140	192.168.20.160	SMTP	C: RCPT TO:
95	192.168.20.140	192.168.20.160	SMTP	C: RCPT TO:
104	192.168.20.140	192.168.20.160	SMTP	C: RCPT TO:

SMTP enumeration example via RCPT TO

OUTLINE

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

▼ 2.2.3 SMTP

2.2.3.1 Malicious SMTP
Traffic Example

2.2.4 DNS

DNS (Domain Name System) is a protocol that resolves names to IP addresses. We will not dive into how DNS works exactly but we'll simply focus on how to recognize normal DNS traffic and suspicious DNS traffic.

You can learn more, or freshen up, on DNS [here](#) and [here](#).

<https://www.ietf.org/rfc/rfc1034.txt>
<https://www.ietf.org/rfc/rfc1035.txt>

IHRPv1 - Caendra Inc. © 2018 | p.162

OUTLINE

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

▼ 2.2.3 SMTP

2.2.3.1 Malicious SMTP
Traffic Example

▼ 2.2.4 DNS

2.2.4 DNS

A few things to point out regarding **DNS (Domain Name System)**:

- DNS is a query-response protocol.
- DNS traffic normally uses UDP on port 53.
- DNS traffic should go to DNS servers only.

OUTLINE

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

▼ 2.2.3 SMTP

2.2.3.1 Malicious SMTP
Traffic Example

▼ 2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

Some facts to help distinguish normal and suspicious DNS traffic

Normal DNS Traffic	Suspicious DNS Traffic
Port 53, UDP	Traffic on port 53 but using TCP instead of UDP.
Should only go to DNS Servers.	DNS traffic not going to DNS Servers.
Should see DNS Responses to DNS Queries.	A lot of DNS Queries with no DNS responses or vice versa.



OUTLINE

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

▼ 2.2.3 SMTP

2.2.3.1 Malicious SMTP
Traffic Example

▼ 2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

Here we see 4 packets: 2 packets for DNS Queries and 2 for DNS Responses.

No.	Time	Source	Destination	Protocol	Length	Info
18	26.200151138	172.16.5.100	172.16.5.10	DNS	83	Standard query 0xde40 PTR 5.5.16.172.in-addr.arpa
19	26.272989431	172.16.5.10	172.16.5.100	DNS	127	Standard query response 0xde40 PTR 5.5.16.172.in-addr.arpa PTR wkst-techsupport.sportsfoo.com
41	56.605405613	172.16.5.100	172.16.5.10	DNS	94	Standard query 0xa620 PTR 5.5.16.172.in-addr.arpa OPT
42	56.639661726	172.16.5.10	172.16.5.100	DNS	136	Standard query response 0xa620 PTR 5.5.16.172.in-addr.arpa PTR wkst-techsupport.sportsfoo.com OPT

Also notice that there are 2 different transaction IDs for the packets: 0xde40 & 0xa620.

OUTLINE

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

▼ 2.2.3 SMTP

2.2.3.1 Malicious SMTP
Traffic Example

▼ 2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

Below we can verify that this is a UDP packet and it's using an expected port, 53.

```
▶ Frame 16: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface 0
▶ Ethernet II, Src: b2:fe:ed:db:02:32 (b2:fe:ed:db:02:32), Dst: vmware_a1:a4:5f (00:50:56:a1:a4:5f)
▶ Internet Protocol Version 4, Src: 172.16.5.100, Dst: 172.16.5.10
▶ User Datagram Protocol, Src Port: 42653, Dst Port: 53
▼ Domain Name System (query)
  [Response in: 19]
  Transaction ID: 0xde40
  ▶ Flags: 0x0100 Standard query
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
  ▼ Queries
    ▼ 5.5.16.172.in-addr.arpa: type PTR, class IN
      Name: 5.5.16.172.in-addr.arpa
      [Name Length: 23]
0000  00 50 56 a1 a4 5f b2 fe ed db 02 32 08 00 45 00  .P.V... ..2..E.
0010  00 45 ae 11 00 00 40 11 6a 08 ac 10 05 64 ac 10  .E...@. j....d..
0020  05 0a a6 9d 00 35 00 31 2d 00 de 40 01 00 00 01  .....5.1 ~..@....
0030  00 00 00 00 00 00 01 35 01 35 02 31 36 03 31 37  .....5 .5.16.17
0040  32 07 69 6e 2d 61 64 64 72 04 61 72 70 61 00 00  2.in-add r.arpa..
0050  0c 00 01
```

OUTLINE

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

▼ 2.2.3 SMTP

2.2.3.1 Malicious SMTP
Traffic Example

▼ 2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

This is the DNS Response to the DNS Query in packet #16. All looks normal as far as the DNS protocol is concerned. A question you would ask yourself is would this be a recon scan or normal behavior within your network?

```
▶ Frame 19: 127 bytes on wire (1016 bits), 127 bytes captured (1016 bits) on interface 0
▶ Ethernet II, Src: vmware_a1:a4:5f (00:50:56:a1:a4:5f), Dst: b2:fe:ed:db:02:82 (b2:fe:ed:db:02:82)
▶ Internet Protocol version 4, Src: 172.16.5.10, Dst: 172.16.5.100
▶ User Datagram Protocol, Src Port: 53, Dst Port: 42653
▼ Domain Name System (response)
  [Request ID: 16]
  [Time: 0.072829293 seconds]
  Transaction ID: 0xde40
  ▶ Flags: 0x8580 Standard query response, No error
  Questions: 1
  Answer RRs: 1
  Authority RRs: 0
  Additional RRs: 0
  ▼ Queries
    ▼ 5.5.16.172.in-addr.arpa: type PTR, class IN
      Name: 5.5.16.172.in-addr.arpa
      [Name Length: 23]
      [Label Count: 6]
      Type: PTR (domain name Pointer) (12)
      Class: IN (0x0001)
  ▼ Answers
    ▼ 5.5.16.172.in-addr.arpa: type PTR, class IN, wkst-techsupport.sportsfoo.com
      Name: 5.5.16.172.in-addr.arpa
      Type: PTR (domain name Pointer) (12)
      Class: IN (0x0001)
      Time to live: 3600
      Data length: 32
      Domain Name: wkst-techsupport.sportsfoo.com
```

OUTLINE

2.2.3 SMTP

2.2.3 SMTP

2.2.3 SMTP

▼ 2.2.3 SMTP

2.2.3.1 Malicious SMTP
Traffic Example

▼ 2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

Here is another packet capture displaying various DNS queries but you see the pattern of DNS Query followed by DNS Response.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.170.8	192.168.170.20	DNS	70	Standard query 0x1032 TXT google.com
2	0.000530	192.168.170.20	192.168.170.8	DNS	98	Standard query response 0x1032 TXT google.com TXT
3	4.005222	192.168.170.8	192.168.170.20	DNS	70	Standard query 0xf76f MX google.com
4	4.837355	192.168.170.20	192.168.170.8	DNS	298	Standard query response 0xf76f MX google.com MX 40 smtp4.google.com MX 10 smtp5.google.com
5	12.817185	192.168.170.8	192.168.170.20	DNS	70	Standard query 0x49a1 LOC google.com
6	12.956209	192.168.170.20	192.168.170.8	DNS	70	Standard query response 0x49a1 LOC google.com
7	20.824827	192.168.170.8	192.168.170.20	DNS	85	Standard query 0x9bbb PTR 104.9.192.66.in-addr.arpa
8	20.825333	192.168.170.20	192.168.170.8	DNS	129	Standard query response 0x9bbb PTR 104.9.192.66.in-addr.arpa PTR 66-192-9-104.gen.twteleco.
9	92.189905	192.168.170.8	192.168.170.20	DNS	74	Standard query 0x75c0 A www.netbsd.org
10	92.238816	192.168.170.20	192.168.170.8	DNS	98	Standard query response 0x75c0 A www.netbsd.org A 204.152.190.12
11	100.965135	192.168.170.8	192.168.170.20	DNS	74	Standard query 0xf0d4 AAAA www.netbsd.org
12	100.202803	192.168.170.20	192.168.170.8	DNS	102	Standard query response 0xf0d4 AAAA www.netbsd.org AAAA 2001:4f8:4:7:2e0:81ff:fe52:9a6b
13	169.027394	192.168.170.8	192.168.170.20	DNS	74	Standard query 0x7f39 AAAA www.netbsd.org
14	169.027781	192.168.170.20	192.168.170.8	DNS	102	Standard query response 0x7f39 AAAA www.netbsd.org AAAA 2001:4f8:4:7:2e0:81ff:fe52:9a6b

Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
Ethernet II, Src: AsustekI_b1:0c:ad (00:e0:18:b1:0c:ad), Dst: QuantaCo_32:41:8c (00:c0:9f:32:41:8c)
Internet Protocol Version 4, Src: 192.168.170.8, Dst: 192.168.170.20
User Datagram Protocol, Src Port: 32795, Dst Port: 53
Domain Name System (query)

OUTLINE

2.2.3 SMTP

2.2.3 SMTP

▼ 2.2.3 SMTP

2.2.3.1 Malicious SMTP
Traffic Example

▼ 2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

▼ 2.2.4 DNS

2.2.4.1 Malicious DNS Traffic Example

A good example of usually suspicious DNS traffic is DNS Zone Transfers.

DNS zone transfers, are a way to replicate DNS databases across a group of DNS servers.

[Click to go back to slide 152](#)

IHRPv1 - Caendra Inc. © 2018 | p.169

OUTLINE

2.2.3 SMTP

▼ 2.2.3 SMTP

2.2.3.1 Malicious SMTP
Traffic Example

▼ 2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

▼ 2.2.4 DNS

▼ 2.2.4.1 Malicious DNS
Traffic Example

2.2.4.1 Malicious DNS Traffic Example

While DNS records are not sensitive individually, if an attacker manages to obtain a copy of the entire DNS zone for a domain (for example by means of an attacker-initiated AXFR request), they may obtain a complete listing of all hosts in that zone.

OUTLINE

▼ 2.2.3 SMTP

2.2.3.1 Malicious SMTP Traffic Example

▼ 2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

▼ 2.2.4 DNS

▼ 2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

Detection

Here we see 3 packets & all packets share one transaction ID. We also see some AXFR which let's us know that this DNS Query is pertaining to a DNS Zone Transfer.

60	82.399428543	172.16.5.100	172.16.5.10	DNS	110 Standard query 0xfc66 AXFR sportsfoo.com OPT
61	82.434731625	172.16.5.10	172.16.5.100	DNS	172 Standard query response 0xfc66 AXFR sportsfoo.com SOA els-winsen2003.sportsfoo.com OPT
63	82.469319332	172.16.5.10	172.16.5.100	DNS	598 Standard query response 0xfc66 AXFR sportsfoo.com SOA els-winsen2003.sportsfoo.com

Let's take a closer look at the packet details for this zone transfer.

OUTLINE

2.2.3.1 Malicious SMTP
Traffic Example

▼ 2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

▼ 2.2.4 DNS

▼ 2.2.4.1 Malicious DNS
Traffic Example

2.2.4.1 Malicious
DNS Traffic Example

2.2.4.1 Malicious
DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

Detection

Here we see details of packet #60, which is a zone transfer. We also see that instead of using UDP/53 this DNS Query used TCP/53. Also in case you didn't know that AXFR referred to a zone transfer the clue is within the packet details.

So why was TCP used instead of UDP? This occurs when the response to a DNS query is too large to fit within a single UDP packet. So the query is resubmitted via TCP to retrieve the entire contents of the response. DNS Responses for zone transfer occur over TCP/53.

```
▶ Frame 60: 110 bytes on wire (880 bits), 110 bytes captured (880 bits) on interface 0
▶ Ethernet II, Src: b2:fe:ed:db:02:32 (b2:fe:ed:db:02:32), Dst: vmware_a1:a4:5f (00:50:56:a1:a4:5f)
▶ Internet Protocol Version 4, Src: 172.16.5.100, Dst: 172.16.5.0
▶ Transmission Control Protocol, Src Port: 58595, Dst Port: 53, Seq: 1, Ack: 1, Len: 44
  ▶ Domain Name System (query)
    [Response In: 63]
    Length: 42
    Transaction ID: 0xfc66
    ▶ Flags: 0x0020 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  ▶ Queries
    ▶ sportsfoo.com: Type AXFR, Class IN
      Name: sportsfoo.com
      [Name Length: 13]
      [Label Count: 2]
      Type: AXFR (transfer of an entire zone) (252)
      Class: IN (0x0001)
    ▶ Additional records
```

OUTLINE

2.2.4.1 Malicious DNS Traffic Example

▼ 2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

▼ 2.2.4 DNS

▼ 2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

Detection

Here we see details of packet #61, which is the response to the zone transfer.

```
▶ Frame 61: 172 bytes on wire (1376 bits), 172 bytes captured (1376 bits) on interface 0
▶ Ethernet II, Src: Vmware_a1:a4:5f (00:50:56:a1:a4:5f), Dst: b2:fe:ed:db:02:32 (b2:fe:ed:db:02:32)
▶ Internet Protocol Version 4, Src: 172.16.5.10, Dst: 172.16.5.100
▶ Transmission Control Protocol, Src Port: 53, Dst Port: 58595, Seq: 1, Ack: 45, Len: 106
▶ Domain Name System (response)
  (Request in: 00)
  [Time: 0.035383882 seconds]
  Length: 104
  Transaction ID: 0xfc66
  ▶ Flags: 0x0000 Standard query response, No error
  Questions: 1
  Answer RRs: 1
  Authority RRs: 0
  Additional RRs: 1
  ▶ Queries
    ▶ sportsfoo.com: type AXFR, class IN
      Name: sportsfoo.com
      [Name Length: 13]
      [Label Count: 2]
      Type: AXFR (transfer of an entire zone) (252)
      Class: IN (0x0001)
  ▶ Answers
    ▶ sportsfoo.com: type SOA, class IN, mname els-winsen2003.sportsfoo.com
      Name: sportsfoo.com
      Type: SOA (Start Of a zone of Authority) (6)
      Class: IN (0x0001)
      Time to live: 3600
      Data length: 50
      Primary name server: els-winsen2003.sportsfoo.com
      Responsible authority's mailbox: hostmaster.sportsfoo.com
      Serial Number: 19
      Refresh Interval: 900 (15 minutes)
      Retry Interval: 600 (10 minutes)
      Expire limit: 86400 (1 day)
      Minimum TTL: 3600 (1 hour)
  ▶ Additional records
```

OUTLINE

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

▼ 2.2.4 DNS

▼ 2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4 DNS

There are other techniques in which DNS can be used for nefarious purposes, such as **DNS tunnels**. We will cover those later in the course.

OUTLINE

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

▼ 2.2.4 DNS

▼ 2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

IHRP

2.3

Effectively Using Open Source IDS



OUTLINE

2.2.4 DNS

2.2.4 DNS

2.2.4 DNS

▼ 2.2.4 DNS

▼ 2.2.4.1 Malicious DNS
Traffic Example

2.2.4.1 Malicious
DNS Traffic Example

2.2.4.1 Malicious
DNS Traffic Example

2.2.4.1 Malicious
DNS Traffic Example

2.2.4.1 Malicious
DNS Traffic Example

2.2.4 DNS

▼ 2.3 Effectively Using Open Source IDS

2.3 Effectively Using Open Source IDS

Now that we have seen our fair share of packets on the wire, it's time to automate the detection of malicious traffic, by using open source Intrusion Detection Systems.

Specifically, we will cover Snort, Bro & Suricata through a series of extensive, in-depth and hands-on labs.

OUTLINE

2.2.4 DNS

2.2.4 DNS

▼ 2.2.4 DNS

▼ 2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4 DNS

▼ 2.3 Effectively Using Open Source IDS

2.3 Effectively Using Open Source IDS

2.3 Effectively Using Open Source IDS

During those labs, you will not only get familiar with their detection capabilities, but you will also learn effective rule writing and how to detect whole classes of attacks.

OUTLINE

2.2.4 DNS

▼ 2.2.4 DNS

▼ 2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4 DNS

▼ 2.3 Effectively Using Open Source IDS

2.3 Effectively Using Open Source IDS

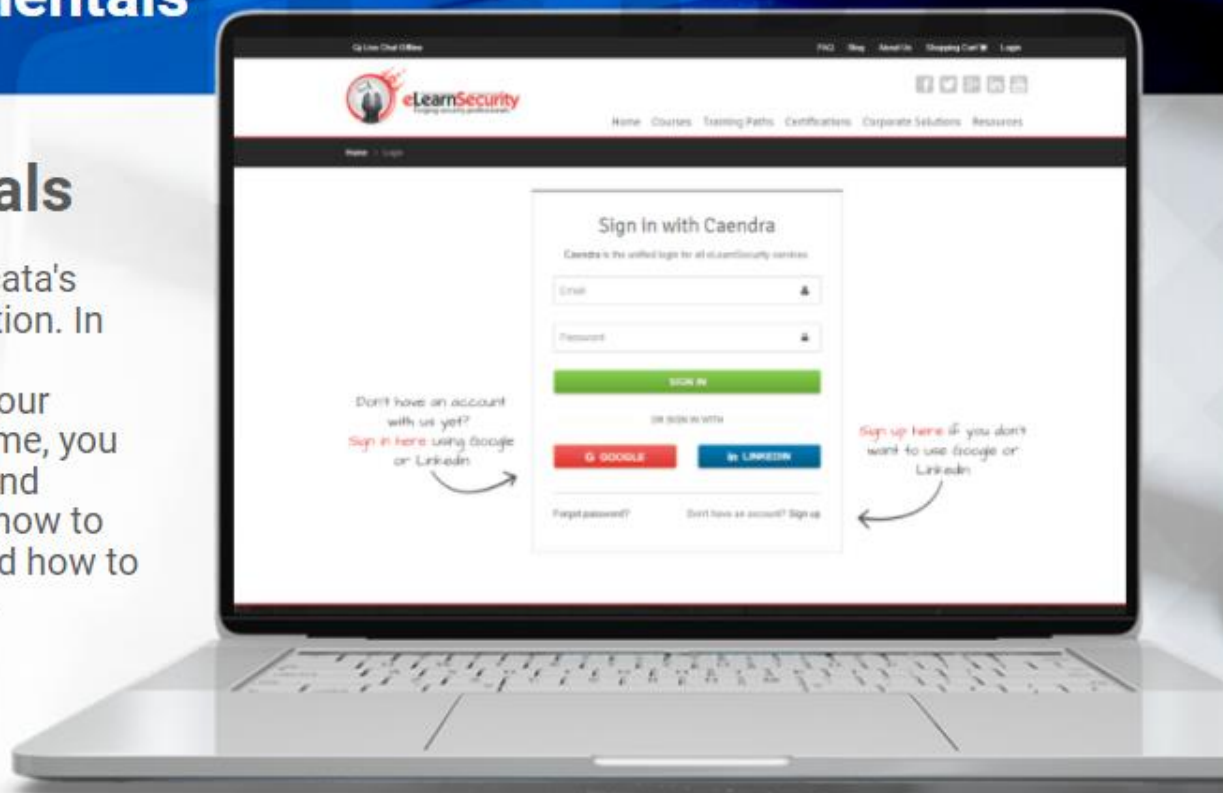
2.3 Effectively Using Open Source IDS

LAB: Suricata Fundamentals

Suricata Fundamentals

In this lab, you will learn about Suricata's capabilities, features and configuration. In addition, you will get familiar with configuring Suricata, according to your detection needs and, at the same time, you will learn all about Suricata Inputs and Outputs. Finally, you will be shown, how to effectively parse Suricata output and how to extract critical information out of it.

**To access, go to the course in your members area and click the lab drop-down in the appropriate module line to access a lab, or go to the virtual labs tab on the left navigation.*



IHRPv1 - Caendra Inc. © 2018 | p.178

OUTLINE

▼ 2.2.4 DNS

▼ 2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4 DNS

▼ 2.3 Effectively Using Open Source IDS

2.3 Effectively Using Open Source IDS

2.3 Effectively Using Open Source IDS

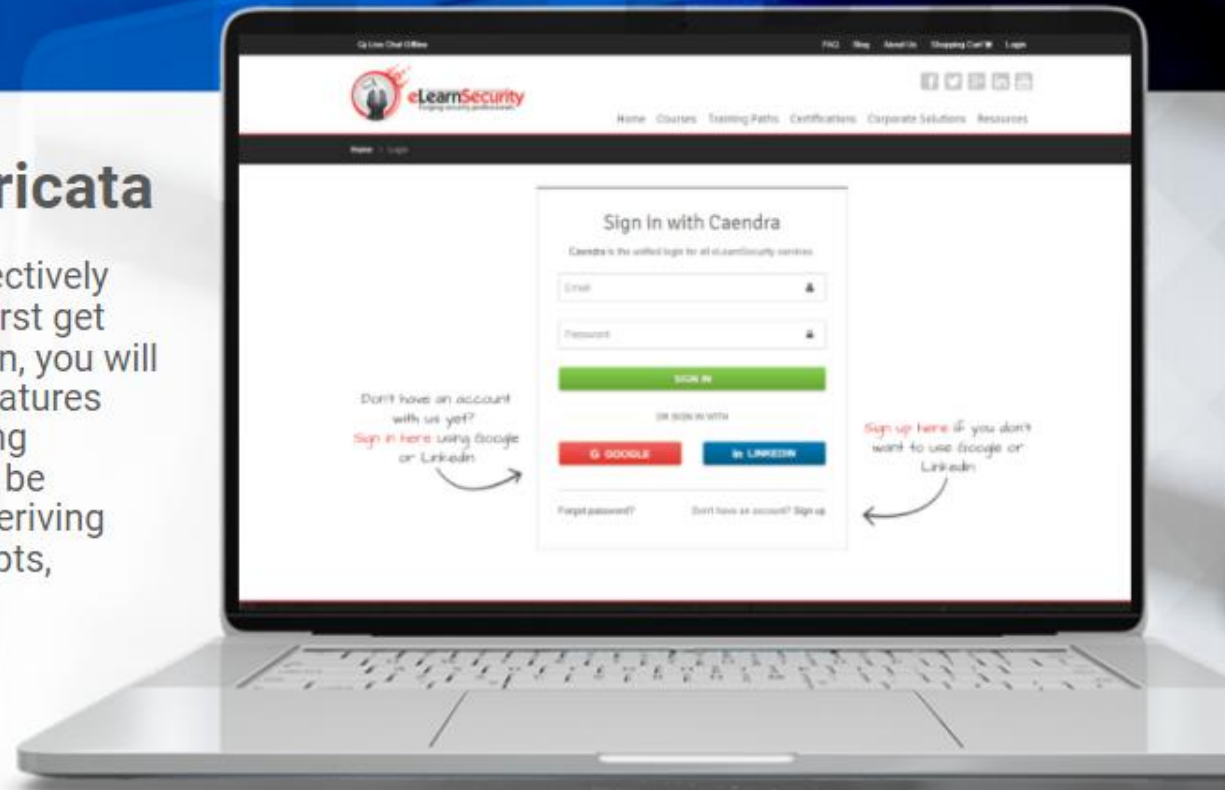
LAB: Suricata Fundamentals

LAB: Effectively Using Suricata

Effectively Using Suricata

In this lab, you will learn how to effectively use Suricata. Specifically, you will first get familiar with Suricata rules, and then, you will learn how to develop your own signatures after analyzing PCAP files containing malicious traffic. Suricata rules will be written to detect malicious traffic deriving from Ransomware, Phishing Attempts, Trojans and Malicious Documents.

**To access, go to the course in your members area and click the lab drop-down in the appropriate module line to access a lab, or go to the virtual labs tab on the left navigation.*



IHRPv1 - Caendra Inc. © 2018 | p.179

OUTLINE

- ▼ 2.2.4.1 Malicious DNS Traffic Example
 - 2.2.4.1 Malicious DNS Traffic Example
 - 2.2.4.1 Malicious DNS Traffic Example
 - 2.2.4.1 Malicious DNS Traffic Example
 - 2.2.4.1 Malicious DNS Traffic Example

2.2.4 DNS

- ▼ 2.3 Effectively Using Open Source IDS
 - 2.3 Effectively Using Open Source IDS
 - 2.3 Effectively Using Open Source IDS

LAB: Suricata Fundamentals

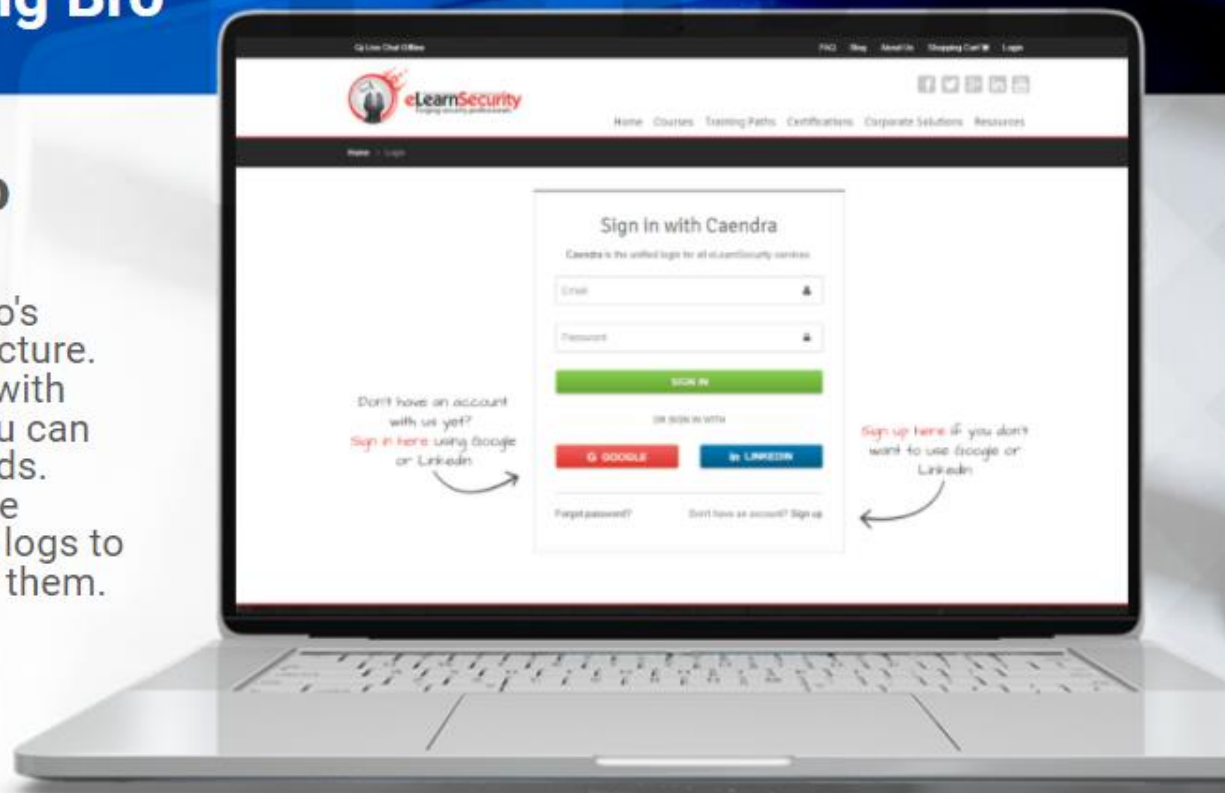
LAB: Effectively Using Suricata

LAB: Effectively Using Bro

Effectively Using Bro

In this lab, you will learn about Bro's capabilities, features, and architecture. Additionally, you will get familiar with effective Bro scripting, so that you can make Bro suit your detection needs. Finally, you will be shown effective manipulation and analysis of Bro logs to extract critical information out of them.

**To access, go to the course in your members area and click the lab drop-down in the appropriate module line to access a lab, or go to the virtual labs tab on the left navigation.*



IHRPv1 - Caendra Inc. © 2018 | p.180

OUTLINE

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4 DNS

▼ 2.3 Effectively Using Open Source IDS

2.3 Effectively Using Open Source IDS

2.3 Effectively Using Open Source IDS

LAB: Suricata Fundamentals

LAB: Effectively Using Suricata

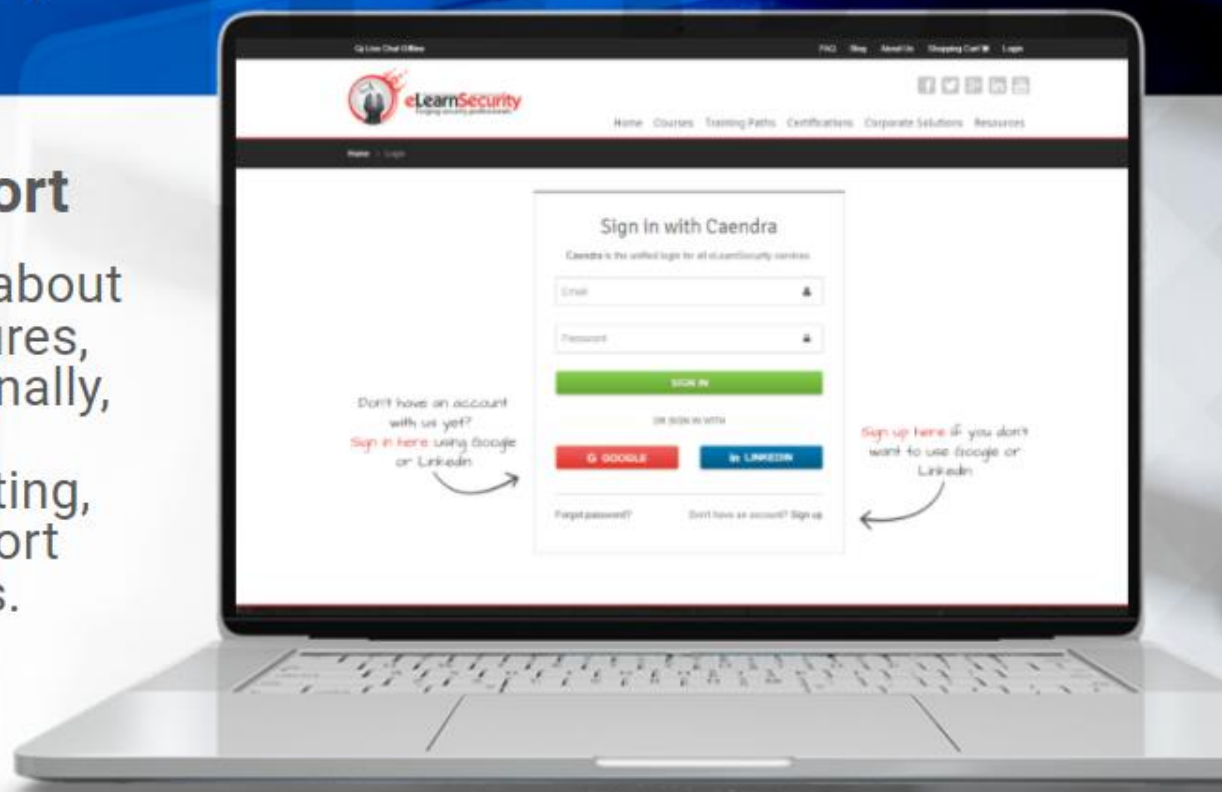
LAB: Effectively Using Bro

LAB: Effectively Using Snort

Effectively Using Snort

In this lab, you will learn about Snort's capabilities, features, and architecture. Additionally, you will get familiar with effective Snort rule scripting, so that you can make Snort suit your detection needs.

**To access, go to the course in your members area and click the lab drop-down in the appropriate module line to access a lab, or go to the virtual labs tab on the left navigation.*



IHRPv1 - Caendra Inc. © 2018 | p.181

OUTLINE

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4 DNS

▼ 2.3 Effectively Using Open Source IDS

2.3 Effectively Using Open Source IDS

2.3 Effectively Using Open Source IDS

LAB: Suricata Fundamentals

LAB: Effectively Using Suricata

LAB: Effectively Using Bro

LAB: Effectively Using Snort

IHRP

References

OUTLINE

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4.1 Malicious DNS Traffic Example

2.2.4 DNS

▼ 2.3 Effectively Using Open Source IDS

2.3 Effectively Using Open Source IDS

2.3 Effectively Using Open Source IDS

LAB: Suricata Fundamentals

LAB: Effectively Using Suricata

LAB: Effectively Using Bro

LAB: Effectively Using Snort

▼ References





TCP protocol

<https://tools.ietf.org/html/rfc793>

Online Wireshark Wiki

https://wiki.wireshark.org/TCP_Relative_Sequence_Numbers

OS fingerprinting

<https://nmap.org/book/osdetect-methods.html#osdetect-gcd>

Slipping in the Window: TCP Reset attacks

https://packetstormsecurity.com/files/download/33170/SlippingInTheWindow_v1.0.doc

References

OUTLINE

2.2.4 DNS Traffic Example

2.2.4.1 Malicious
DNS Traffic Example

2.2.4 DNS

▼ 2.3 Effectively Using Open Source IDS

2.3 Effectively Using Open Source
IDS

2.3 Effectively Using Open Source
IDS

LAB: Suricata Fundamentals

LAB: Effectively Using Suricata

LAB: Effectively Using Bro

LAB: Effectively Using Snort

▼ References

References





References

TCP Timestamps option

<https://tools.ietf.org/html/rfc1323>

Remote Physical Device Fingerprinting

<http://www.caida.org/publications/papers/2005/fingerprinting/KohnoBroidoClaffy05-devicefingerprinting.pdf>

Misusing TCP Timestamps

<https://www.scip.ch/en/?labs.20150305>

Remote OS detection via TCP/IP Stack FingerPrinting

<https://nmap.org/nmap-fingerprinting-article.txt>

OUTLINE

2.2.3 Remote OS Detection

2.2.4 DNS

▼ 2.3 Effectively Using Open Source IDS

2.3 Effectively Using Open Source IDS

2.3 Effectively Using Open Source IDS

LAB: Suricata Fundamentals

LAB: Effectively Using Suricata

LAB: Effectively Using Bro

LAB: Effectively Using Snort

▼ References

References

References





ICMP Usage in Scanning

https://www.cs.dartmouth.edu/~sergey/me/netreads/ICMP_Scanning_v3.0.pdf

Ptunnel

<http://freshmeat.sourceforge.net/projects/ptunnel/>

ONC RPC

<https://www.ietf.org/rfc/rfc1831.txt>

DCE RPC

<http://pubs.opengroup.org/onlinepubs/9629399/>

References

OUTLINE

▼ 2.3 Effectively Using Open Source IDS

2.3 Effectively Using Open Source IDS

2.3 Effectively Using Open Source IDS

LAB: Suricata Fundamentals

LAB: Effectively Using Suricata

LAB: Effectively Using Bro

LAB: Effectively Using Snort

▼ References

References

References

References





RFC 2616

<https://www.ietf.org/rfc/rfc2616.txt>

SQL injection

<https://www.acunetix.com/websecurity/sql-injection/>

Sqlmap

<https://github.com/sqlmapproject/sqlmap>

RFC 6101

<https://tools.ietf.org/html/rfc6101>

References

OUTLINE

2.3 Effectively Using Open Source IDS

2.3 Effectively Using Open Source IDS

LAB: Suricata Fundamentals

LAB: Effectively Using Suricata

LAB: Effectively Using Bro

LAB: Effectively Using Snort

▼ References

References

References

References

References





[RFC 5246](#)

<https://tools.ietf.org/html/rfc5246>

[RFC 8446](#)

<https://datatracker.ietf.org/doc/rfc8446/>

[SSL Renegotiation Attack](#)

<https://blog.qualys.com/ssllabs/2011/10/31/tls-renegotiation-and-denial-of-service-attacks>

[Domain names - concepts and facilities](#)

<https://www.ietf.org/rfc/rfc1034.txt>

References

OUTLINE

1.2.2

2.3 Effectively Using Open Source
IDS

LAB: Suricata Fundamentals

LAB: Effectively Using Suricata

LAB: Effectively Using Bro

LAB: Effectively Using Snort

▼ References

References

References

References

References

References



References

Domain names - implementation and specification

<https://www.ietf.org/rfc/rfc1035.txt>

Transaction IDs

[https://technet.microsoft.com/en-us/library/dd197470\(v=ws.10\).aspx](https://technet.microsoft.com/en-us/library/dd197470(v=ws.10).aspx)

OUTLINE

LAB: Suricata Fundamentals

LAB: Effectively Using Suricata

LAB: Effectively Using Bro

LAB: Effectively Using Snort

▼ References

References

References

References

References

References

References

Labs

Suricata Fundamentals

In this lab, you will learn about Suricata's capabilities, features and configuration. In addition, you will get familiar with configuring Suricata, according to your detection needs and, at the same time, you will learn all about Suricata Inputs and Outputs. Finally, you will be shown, how to effectively parse Suricata output and how to extract critical information out of it.

Effectively Using Suricata

In this lab, you will learn how to effectively use Suricata. Specifically, you will first get familiar with Suricata rules, and then, you will learn how to develop your own signatures after analyzing PCAP files containing malicious traffic. Suricata rules will be written to detect malicious traffic deriving from Ransomware, Phishing Attempts, Trojans and Malicious Documents.

**To access, go to the course in your members area and click the lab drop-down in the appropriate module line to access a lab, or go to the virtual labs tab on the left navigation.*

OUTLINE

LAB: Effectively Using Suricata

LAB: Effectively Using Bro

LAB: Effectively Using Snort

▼ References

References

References

References

References

References

References

Labs

Labs

Effectively Using Bro

In this lab, you will learn about Bro's capabilities, features, and architecture. Additionally, you will get familiar with effective Bro scripting, so that you can make Bro suit your detection needs. Finally, you will be shown effective manipulation and analysis of Bro logs to extract critical information out of them.

Effectively Using Snort

In this lab, you will learn about Snort's capabilities, features, and architecture. Additionally, you will get familiar with effective Snort rule scripting, so that you can make Snort suit your detection needs.

**To access, go to the course in your members area and click the lab drop-down in the appropriate module line to access a lab, or go to the virtual labs tab on the left navigation.*

OUTLINE

LAB: Effectively Using Bro

LAB: Effectively Using Snort

▼ References

References

References

References

References

References

References

Labs

Labs