



Wireshark

By: Kamil Chaudhry, Asad Khan, Hamza Mehmood

The Agenda

- **CSC347 Review**
- **Tools & Filters**
- **Advanced Use Cases**
- **Practical Demonstrations**
- **Real-Life Examples**



First, A Quick Story...

Content Warning (Sensitive Topics)

Source: [The School Shootings Were Fake. The Terror was Real](#) (very interesting stuff, read the whole thing if you have time)





Brad Dennis

- Private investigator hired by twitch streamers
- Uses detective skills to join a private telegram channel
- Joins a WoW guild with member named Nazgul (AKA Torswats)
- Notices Nazgul's telegram switched from swatting twitch streamers to schools



Nazgul's Arrest

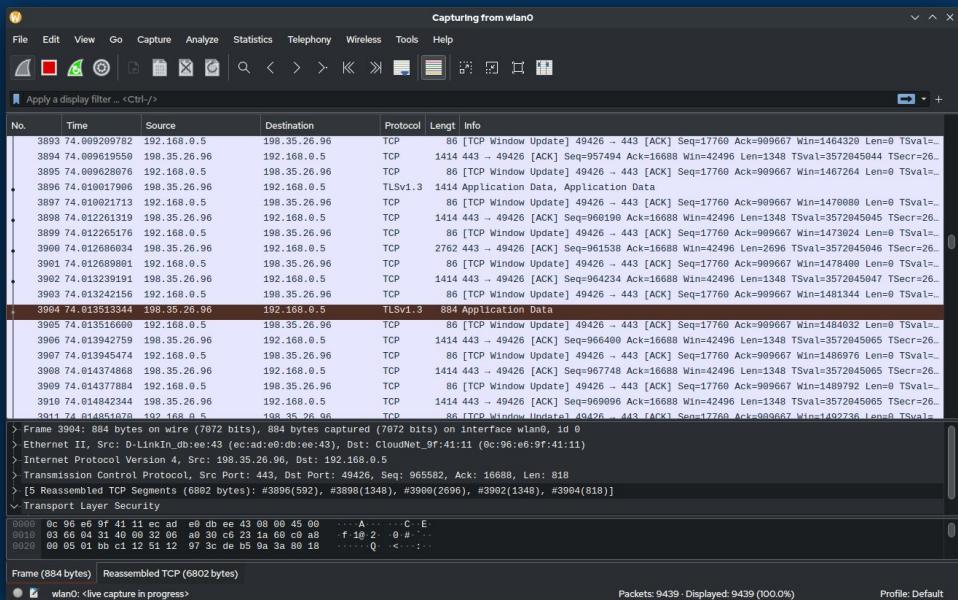
- Brad convinced Nazgul to add him as a friend on Tox, tempting him with a business opportunity
- Tox uses a two-way IP to IP tunnel
- Brad starts a wireshark capture as he communicates with Nazgul
- Successfully creates a packet capture file containing Nazgul's IP address
- Provided the FBI with the .pcap file containing the IP in a USB drive
- Nazgul eventually arrested and pleaded guilty



Introduction to Wireshark

Introduction to Wireshark

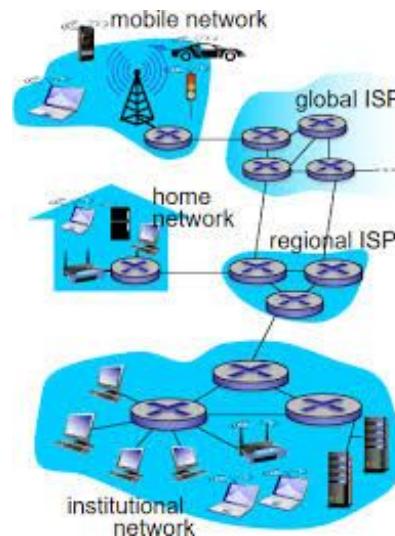
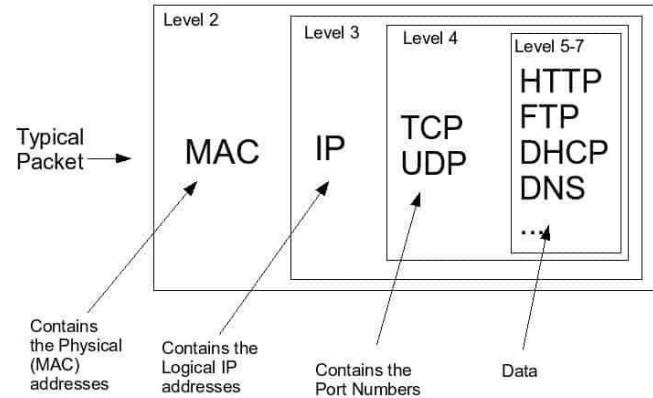
- Open-Source Network Packet/Protocol Analyzer
 - Captures packets and displays data in detail
 - Written in C, C++ and Lua
- Invented by Gerald Combs in 1998
 - Originally named “Ethereal”
 - Later renamed “Wireshark” in 2006
- Used for various purposes
 - Troubleshoot Network Issues
 - Examine Security Issues/Vulnerabilities
 - Debug Protocol Implementations



Example of a Wireshark GUI

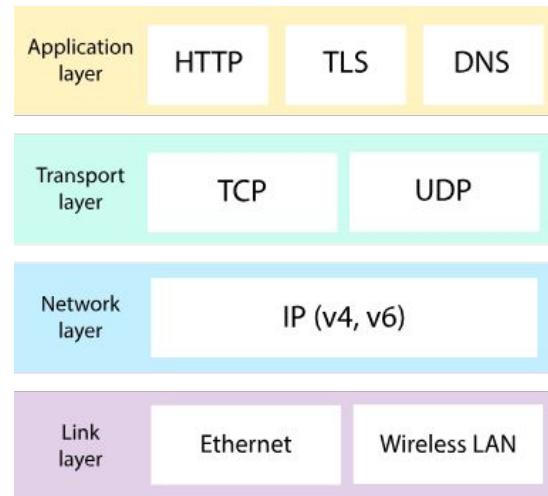
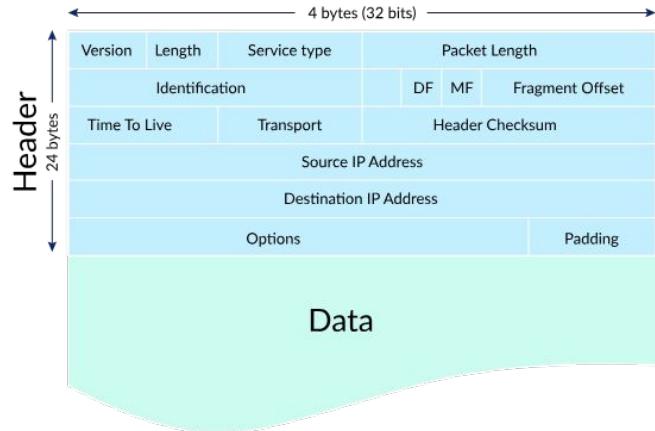
Introduction to Networks

- A system of interconnected devices for communication and data sharing.
- Each device on a network is identified by its IP address
- DNS servers map domain names to IP addresses
- Devices talk to each other by sending packets over the network



Protocols

- Packets contain routing information along with the data
- Packets always follow protocols
- Routing information encapsulating the data (payload)
- Each layer has their own protocols
 - Used to forward or interpret the packet at the receiving end



Wireshark Usage

Installation

- Available on:
<https://wireshark.com/download.html>
- Managed by Wireshark Foundation
- Open-Source
- Cross-Platform:
 - Linux
 - MacOS
 - Windows
- Can be built directly from source as well



Wireshark

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Google_89:0e:c8	Broadcast	ARP	42	Who has 192.168.86.65? Tell 192.168.86.1
2	0.001605	52.202.62.206	192.168.86.65	TLSv1...	85	Encrypted Alert
3	1.022481	Google_89:0e:c8	Broadcast	ARP	42	Who has 192.168.86.65? Tell 192.168.86.1
4	2.046477	Google_89:0e:c8	Broadcast	ARP	42	Who has 192.168.86.65? Tell 192.168.86.1
5	3.696138	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0x56f415ed
6	3.890093	Google_89:0e:c8	Broadcast	ARP	42	Who has 192.168.86.65? Tell 192.168.86.1
7	4.708880	Google_89:0e:c8	Broadcast	ARP	42	Who has 192.168.86.65? Tell 192.168.86.1
8	5.120839	52.202.62.206	192.168.86.65	TLSv1...	85	Encrypted Alert
9	5.285935	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0x56f415ed
10	5.733257	Google_89:0e:c8	Broadcast	ARP	42	Who has 192.168.86.65? Tell 192.168.86.1
11	6.756877	Google_89:0e:c8	Broadcast	ARP	42	Who has 192.168.86.65? Tell 192.168.86.1
12	6.962423	192.168.86.1	192.168.86.65	DHCP	342	DHCP Offer - Transaction ID 0x56f415ed
13	6.962436	192.168.86.1	192.168.86.65	DHCP	342	DHCP Offer - Transaction ID 0x56f415ed
14	6.962437	52.202.62.206	192.168.86.65	TCP	85	[TCP Retransmission] 443 → 54850 [FIN, PSH, ACK] Seq=1 Ack=1 Win=9 Len=31
15	7.372560	52.202.62.206	192.168.86.65	TCP	85	[TCP Retransmission] 443 → 54850 [FIN, PSH, ACK] Seq=1 Ack=1 Win=9 Len=31
16	7.965588	0.0.0.0	255.255.255.255	DHCP	342	DHCP Request - Transaction ID 0x56f415ed
17	7.977178	192.168.86.1	192.168.86.65	DHCP	342	DHCP ACK - Transaction ID 0x56f415ed
18	7.977524	Apple_98:09:27	Broadcast	ARP	42	Who has 192.168.86.65? (ARP Probe)
19	8.191761	52.202.62.206	192.168.86.65	TCP	85	[TCP Retransmission] 443 → 54850 [FIN, PSH, ACK] Seq=1 Ack=1 Win=9 Len=31
20	8.300353	Apple_98:09:27	Broadcast	ARP	42	Who has 192.168.86.65? (ARP Probe)
21	8.625204	Apple_98:09:27	Broadcast	ARP	42	Who has 192.168.86.65? (ARP Probe)
22	8.949203	Apple_98:09:27	Broadcast	ARP	42	ARP Announcement for 192.168.86.65
23	9.270375	Apple_98:09:27	Broadcast	ARP	42	ARP Announcement for 192.168.86.65
24	9.591469	Apple_98:09:27	Broadcast	ARP	42	ARP Announcement for 192.168.86.65
25	9.593415	Apple_98:09:27	Broadcast	ARP	42	Who has 192.168.86.17? Tell 192.168.86.65
26	9.595976	Google_89:0e:c8	Apple_98:09:27	ARP	42	192.168.86.1 is at 3c:28:6d:89:0e:c8
27	9.708931	Apple_98:09:27	Broadcast	ARP	42	Who has 192.168.86.17? Tell 192.168.86.65
28	9.713739	Google_89:0e:c8	Apple_98:09:27	ARP	42	192.168.86.1 is at 3c:28:6d:89:0e:c8
29	9.713795	192.168.86.65	192.168.86.1	DNS	79	Standard query 0xcc80 PTR lb._dns-sd._udp.lan
30	9.713795	192.168.86.65	192.168.86.1	DNS	79	Standard query 0x95c0 PTR db._dns-sd._udp.lan
31	9.713796	192.168.86.65	192.168.86.1	DNS	78	Standard query 0xf0ed PTR b._dns-sd._udp.lan
32	9.713796	192.168.86.65	192.168.86.1	DNS	101	Standard query 0xb3b3 PTR lb._dns-sd._udp.0.86.168.192.in-addr.arpa
33	9.713796	192.168.86.65	192.168.86.1	DNS	101	Standard query 0x53f8 PTR db._dns-sd._udp.0.86.168.192.in-addr.arpa
34	9.713796	192.168.86.65	192.168.86.1	DNS	100	Standard query 0xb4dd PTR b._dns-sd._udp.0.86.168.192.in-addr.arpa
35	9.713797	192.168.86.65	192.168.86.1	DNS	75	Standard query 0x3082 A notify.adobe.io
36	9.713797	192.168.86.65	192.168.86.1	DNS	86	Standard query 0xfaa5 PTR 65.86.168.192.in-addr.arpa
37	9.719044	192.168.86.1	192.168.86.65	DNS	79	Standard query response 0xcc80 No such name PTR lb._dns-sd._udp.lan
38	9.719049	192.168.86.1	192.168.86.65	DNS	79	Standard query response 0x95c0 No such name PTR db._dns-sd._udp.lan
39	9.720131	192.168.86.1	192.168.86.65	DNS	78	Standard query response 0xf0ed No such name PTR b._dns-sd._udp.lan

> Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface en0, id 0
 > Ethernet II, Src: Google_89:0e:c8 (3c:28:6d:89:0e:c8), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 > Address Resolution Protocol (request)

0000	ff	ff	ff	ff	ff	ff	3c	28	6d	89	0e	c8	08	06	00	01<(m.....
0010	08	00	06	04	00	01	3c	28	6d	89	0e	c8	c0	a8	56	01<(m.....V
0020	00	00	00	00	00	00	c0	a8	56	41	VA					

Plugins

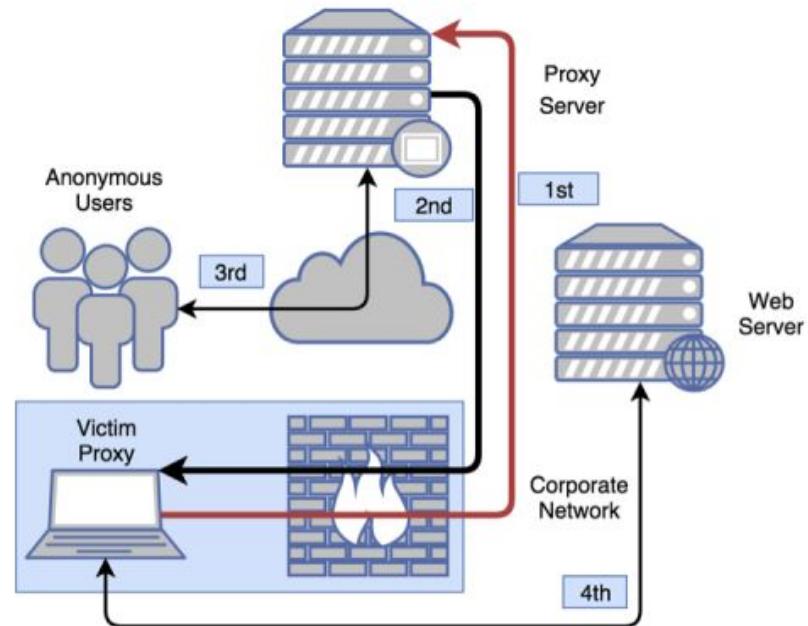
- Wireshark supports custom open-source plugins
- Plugins add extra features for dissecting protocols and interfacing with third-party tools
- Wireshark has a vast community of developers
- Almost always a solution for what you need

No.	Time	Source	Destination	Protocol	Length	Info
127	1.737673	209.222.115.120	192.168.2.29	TCP	66	25565 → 51135 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1332 SACK_PERM=1 WS=512
128	1.737835	192.168.2.29	209.222.115.120	TCP	54	51135 → 25565 [ACK] Seq=1 Ack=1 Win=65280 Len=0
129	1.739195	192.168.2.1	192.168.2.29	DNS	186	Standard query response 0x98e3 A playjava.manacube.net CNAME 188a1d97abba0437744e034745df0f37.ipv4.t
130	1.742750	192.168.2.29	50.114.4.120	TCP	66	51138 → 25565 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
131	1.764776	185.199.95.107	192.168.2.29	TCP	66	25565 → 51136 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1240 SACK_PERM=1 WS=128
132	1.764949	192.168.2.29	185.199.95.107	TCP	54	51136 → 25565 [ACK] Seq=1 Ack=1 Win=65280 Len=0
133	1.774810	50.114.4.120	192.168.2.29	TCP	66	25565 → 51138 [SYN, ACK] Seq=0 Ack=1 Win=42340 Len=0 MSS=1452 SACK_PERM=1 WS=2048
134	1.774995	192.168.2.29	50.114.4.120	TCP	54	51138 → 25565 [ACK] Seq=1 Ack=1 Win=65280 Len=0
135	1.814355	192.168.2.29	50.114.4.120	Minecraft	83	Set Protocol [handshaking] (toServer)
136	1.814355	192.168.2.29	209.222.115.120	Minecraft	75	[Set Protocol [handshaking] (toServer)]
137	1.814679	192.168.2.29	209.222.115.120	Minecraft	56	Ping Start [status1] (toServer)

```
> Frame 136: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface \Device\NPF_{86D1FD1E-E87C-4F03-A0F3-688935FDDED9}, id 0
> Ethernet II, Src: f8:89:d2:16:b8:93 (f8:89:d2:16:b8:93), Dst: 0c:ac:8a:93:48:8a (0c:ac:8a:93:48:8a)
> Internet Protocol Version 4, Src: 192.168.2.29, Dst: 209.222.115.120
> Transmission Control Protocol, Src Port: 51135, Dst Port: 25565, Seq: 1, Ack: 1, Len: 21
> Minecraft Protocol
    Length: 20
    Packet Id: 0x00000000
    Protocol Version: 47
    Server Host: \016mc.hypixel.net
    Server Port: 25565
    Next State: 1
```

Wireshark Usage

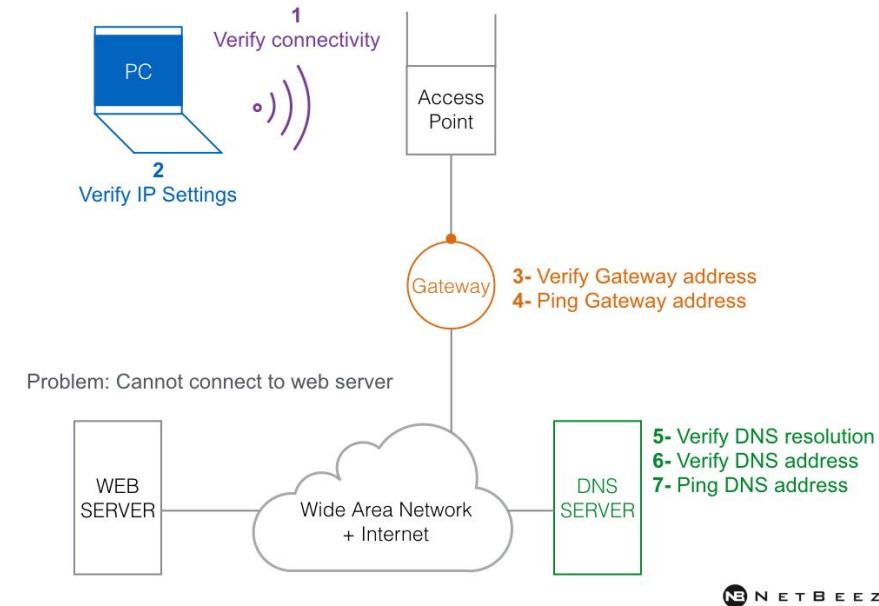
- Network Troubleshooting
 - Identifying slow responses, dropped packets, or bandwidth hogs
 - Analyzing DNS responses to see if requests are timing out
- Incident Response
 - Identifying infected hosts and suspicious traffic patterns
- Malware Analysis
 - Observing communication patterns
 - C2 traffic
 - Suspicious domains



Example Case

Identifying Users on a Network

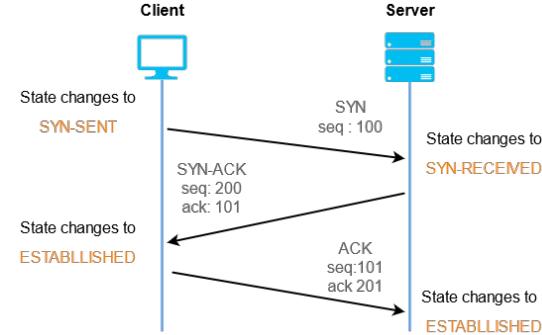
- Users within an organization's network can become compromised from a malicious attack
- Requires a review of packet capture (.pcap) data to analyze network traffic generated by the affected host
- Wireshark can be used to identify host and user data in .pcap files
- Packet capture data sourced from Palo Alto Networks (Unit 42)



Malicious Uses

Man in the Middle Attacks

- 3-way handshake an ISN is established
- Wireshark to detect connection
- Collect ISN then authentication
- Forge request on user or server behalf
- Works in cases where servers are hosting unencrypted traffic
 - i.e HTTP



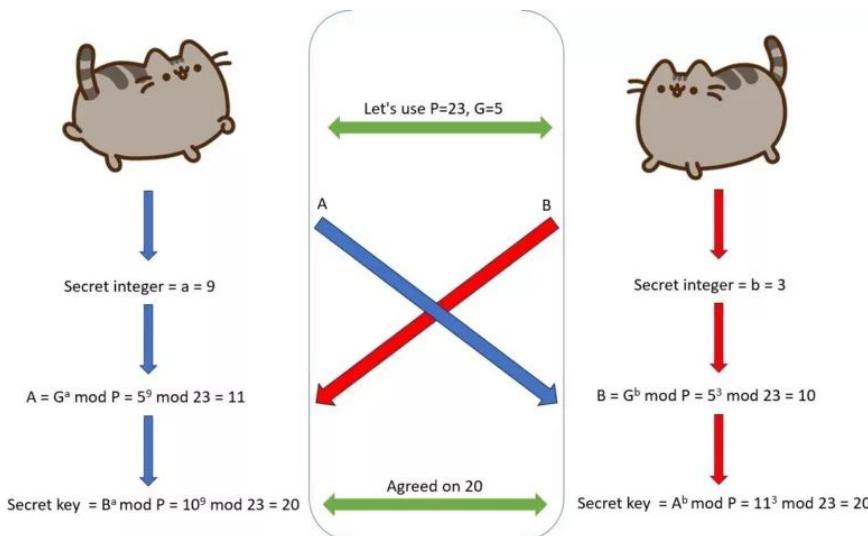
MITM Reassembling Files

- Files being uploaded over insecure protocol
- Can be reassembled after being intercepted
- Sensitive files can potentially be used as blackmail

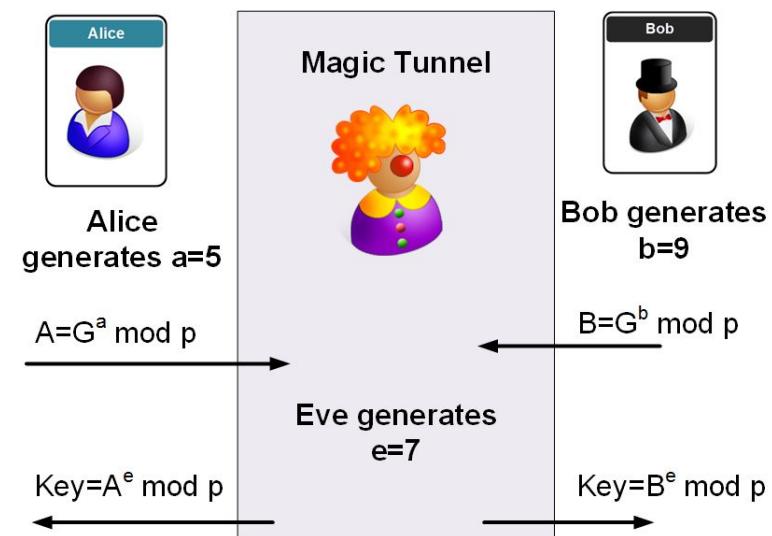


MITM with DHKE

Normal Exchange



Man in the Middle



DNS Spoofing

- DNS requests plain text



```
DNS      75 Standard query 0xf690 A xmpp005.zoom.us
```

- Victim sends DNS request for some address www.example.com
- Attacker is using wireshark and filtering for dns requests and finds one for the website whose replica they have created
- Attacker forges a response for a site replica that they are hosting on another ip address
- Victim puts sensitive data into the replica – their data is collected by attacker

Scapy

- An interactive packet manipulation tool written in Python
- Similar to Wireshark, can be used to sniff and decode packets
- It can also be used to send packets
- Doesn't have a nice GUI like Wireshark
- In the lab you will be using the `scapy` python library to forge a DNS response



Defending Against Snooping & Attacks

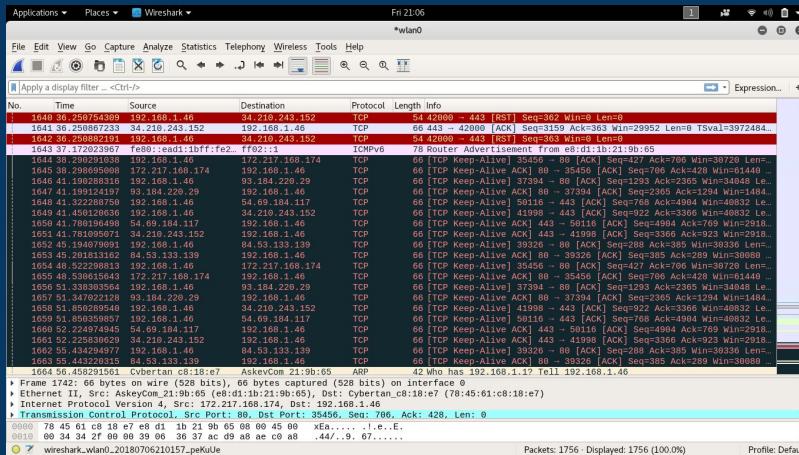
Encryption

- Use encrypted or safe versions of services
 - HTTPS over HTTP
 - SSH over TELNET
 - SFTP over FTP
 - SMTPS over SMTP
- Prevents data from being exposed to snoopers



Vigilance

- Avoid public WiFi
- Use a trusted VPN
- Ensure URL correctness & certificates
- Ensure network speeds are within expected range
- Inspect your own network traffic to identify suspicious requests



Snort

- Quite difficult to always keep track of network activity
- Snort monitors network activity and uses a series of rules that help define malicious network activity
- Generates alerts in case of malicious activity
- Packets capture on snort can be forwarded to Wireshark for inspection

*enp1s0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

snort

No.	Timestamp	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.122.253	8.8.8.8	ICMP	98	Echo (ping) request id=0x0001, seq=11/2816, t
2	0.010187716	8.8.8.8	192.168.122.253	ICMP	98	Echo (ping) reply id=0x0001, seq=11/2816, t
4	1.001736548	192.168.122.253	8.8.8.8	ICMP	98	Echo (ping) request id=0x0001, seq=12/3072, t
5	1.011755529	8.8.8.8	192.168.122.253	ICMP	98	Echo (ping) reply id=0x0001, seq=12/3072, t
6	2.002882793	192.168.122.253	8.8.8.8	ICMP	98	Echo (ping) request id=0x0001, seq=13/3328, t
7	2.013214078	8.8.8.8	192.168.122.253	ICMP	98	Echo (ping) reply id=0x0001, seq=13/3328, t
9	3.0049832413	192.168.122.253	8.8.8.8	ICMP	98	Echo (ping) request id=0x0001, seq=14/3584, t
10	3.015678919	8.8.8.8	192.168.122.253	ICMP	98	Echo (ping) reply id=0x0001, seq=14/3584, t
11	4.007537991	192.168.122.253	8.8.8.8	ICMP	98	Echo (ping) request id=0x0001, seq=15/3840, t
12	4.018379034	8.8.8.8	192.168.122.253	ICMP	98	Echo (ping) reply id=0x0001, seq=15/3840, t

Snort: (msg: "ICMP PING" sid: 384 rev: 5) [from Running Snort]

 [Expert Info (Warning/Security): Alert 384: "ICMP PING"]
 [Raw Alert: 04/17/22-20:12:47.228227 [*] [1:384:5] ICMP PING [*] [Classification: Misc activity] [Priority: 3] {ICMP} 1...]
 [Alert Classification: Misc activity]
 [Rule ICMP PING (sid=384, rev=5)]
 [Alert Message: ICMP PING]
 [Rule SID: 384]
 [Rule Revision: 5]
 [Rule Generator: 1]
 [Alert Priority: 3]
 [Rule String: alert icmp \$EXTERNAL_NET any -> \$HOME_NET any (msg:"ICMP PING"; icode:0; itype:8; classtype:misc-activity;...]
 [Protocol: icmp]

Frame (98 bytes) Rule String (190 bytes) Rule String (123 bytes)

Snort Alerts: Protocol

Packets: 13 · Displayed: 10 (76.9%) · Dropped: 0 (0.0%) · Profile: Default



The Norman Cryptominer

The Case

- Varonis Security Research Team
- Monero crypto mining infection at a mid-sized company
- Discovery of a new cryptocurrency mining variant named “Norman”
- Infection spread to nearly all servers and workstations
- Infection caused major slowdowns and application instability



Norman

- XMRIG-based cryptocurrency mining software
 - Open-Source tool
 - Used for mining Monero
- Monero
 - Cryptocurrency w/ privacy focus
- Payloads injected into legitimate process for stealth
- Terminates the mining process upon Task Manager detection
- Uses legitimate looking processes
 - 'wuapp.exe'



Wireshark's Role in the Case

- Varonis Data Security Platform alerted of suspicious activity
- Abnormal web activity and correlated file behaviours flagged
- Wireshark used to monitor network communications and identify command-and-control patterns
- Application of Wireshark filters found that infected hosts were accessed by DuckDNS
 - DNS service to map dynamic IPs to static domains



Stopping the Attack

- Varonis team could see all IP addresses of the C&C servers the attackers were using with Wireshark.
 - Revealed a PHP shell that continually connects to the C&C server.
 - Found an XSL file being run by a known windows executable.
 - Was actually Zend Guard obfuscated PHP code.
 - Led to finding that the malware's payload is based on the execution in the XSL file.
 - Allowed them to do further analysis by deciphering the XSL file.

Obfuscated XSL File

Conclusions

- Wireshark as a packet analysis tool
- Wireshark GUI & configuration
- Wireshark used for network administration and information security purposes
- Specific attacks and how to address them (Man-in-the-Middle, DNS Spoofing)
- Demonstrated how filters & analysis features can aid in detecting & mitigating threats
- Real-world cases showing Wireshark's effectiveness in identifying and resolving network issues

