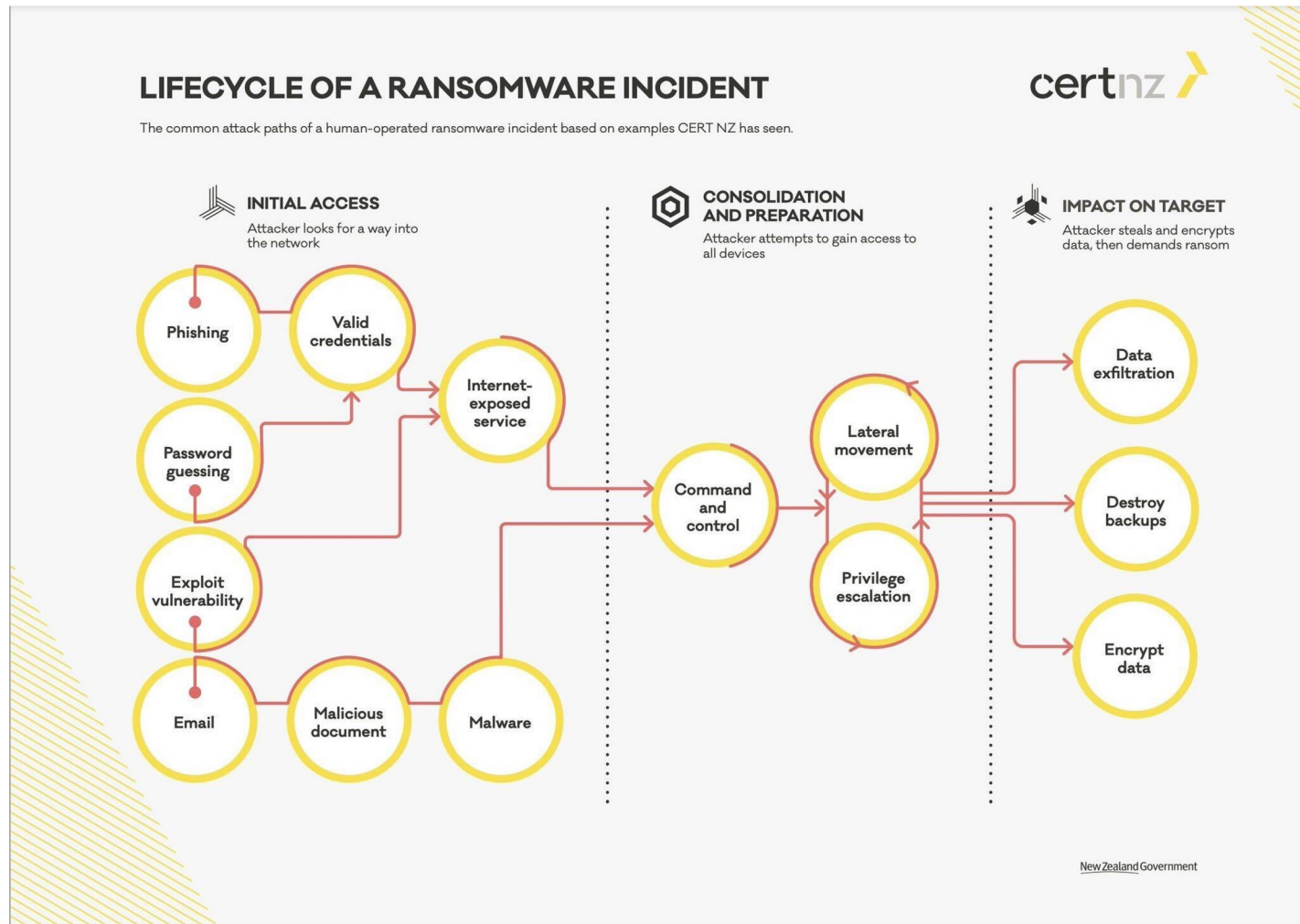


# Network Forensics Introduction

# Typical Attack Path

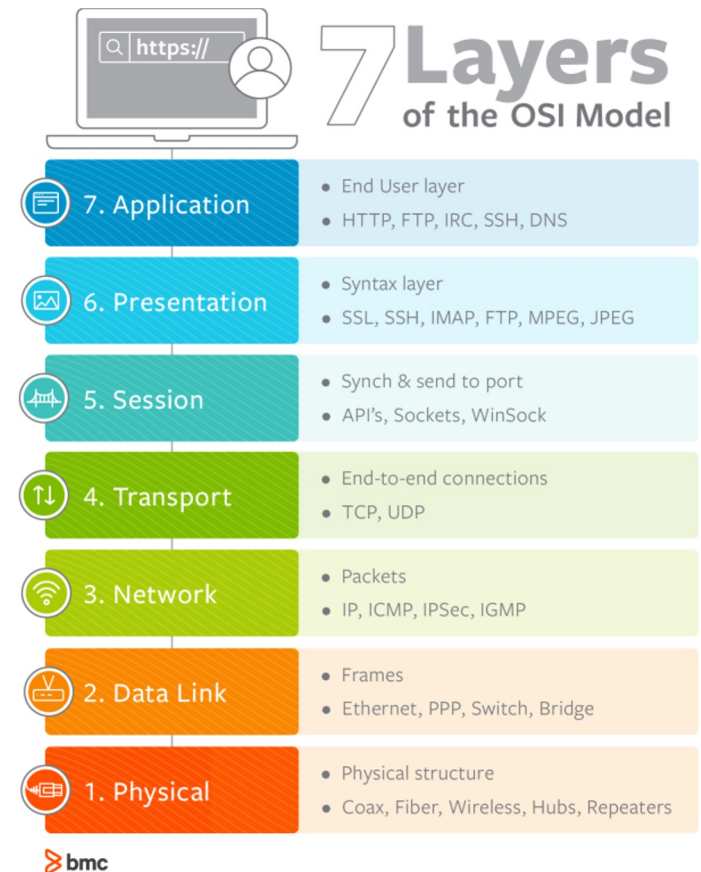


# Networking Basics

- Communications.
- Two systems communicating with one another, typically over a local network (LAN) or the Internet.
- Variety of messaging protocols, both stateful and stateless depending on the type of message.

# Networking Basics

- Encapsulation
- OSI Model
- Encryption
- Compression



# TCP / UDP Refresher

- Transport-layer protocols.
- Learn your ports.
- Typically the most interesting forensically because these are the rough level of encapsulation where we're watching connections between systems.
- Transmission Control Protocol
  - Stateful
  - Examples include: HTTP (web), SMTP/IMAP (mail), DNS (sometimes)
- User Datagram Protocol
  - Stateless
  - Examples include: DNS (most of the time)

# Network Forensics is Hard

- Encryption
- Compression
- Volume of data.
- Amount of noise.
- Encryption.

# Protocol Definition

- Request for Comments (RFC)
  - Internet Engineering Task Force (IETF)
  - Document that tracks technical protocol specifications.
  - Process / context lives in RFCs
- Key RFCs for networking:
  - RFC 1918 (private networking)
  - RFC 2616 / 7230 (HTTP 1.1)
  - RFC 7540 (HTTP v2)

# IP Layer – How it Works

- IP address (192.168.8.8)
  - Source
  - Destination
- IPv4
  - 4 digits 0-255
- IPv6
  - 8 groups of 4 hex digits
  - More structured than IPv4
- NAT
  - Internal / external IP scheme
  - Internally will see internal IPs
  - Externally will see single gateway IP





# Subnetting

- CIDR Notation
  - “/”
  - Uses a subnet mask
- Subnets
  - Separation / segmentation of IP networks
- 192.168.1.0-256 /24 (256)
- /16 (65,536)
- /8 (16,777,216)

# IP Layer – Key Information

- Loopback
  - 127.0.0.1
- All addresses on the machine
  - 0.0.0.0
- RFC 1918
  - 10.0.0.0-10.255.255.255 (10.0.0.0/8)
  - 172.16.0.0/12
  - 192.168.0.0/16

# A Normal Day in Networkland

- Type a URL (domain) + hit enter
- DNS request for domain.com
- DNS response for domain.com
- HTTP GET request for domain.com
- HTTP 200 response with content for domain.com

# Capturing Network Forensics Data

- Capturing network data either requires a dedicated (and pre-positioned) network tap.
- A network allows for a copy of all traffic coming and going (RX and TX) to be sent to an additional interface.
- A capture interface can be leveraged to get access to process or capture traffic.



# Network Forensic Capture Cont'd

- Pros:
  - Full capture of everything.
  - Can include files, non-standard protocols, and a lot more.
- Cons:
  - Typically have to decrypt in-line / MITM traffic.
  - Newer TLS versions are making “passive” decryption difficult.
  - Encryption not always possible (TLS1.3)
  - Harder and harder as people move to cloud environments.

# Network Metadata Capture

- Capture is becoming less and less feasible due to data transmission and storage limitations.
  - $100 \text{ MBPS} \times 7\text{d} = 7.56\text{TB}$
  - $10 \text{ GBPS} \times 7\text{d} = 756\text{TB}$
  - $10 \text{ GBPS} \times 30\text{d} = 3.26\text{PB}$
- Pros
  - Fast
  - Low storage requirements
- Cons
  - Processing overhead is also a challenge without hardware offloading and specialized drivers (like AF\_PACKET which allow raw packet access).

# Network Metadata Capture Cont'd

## Network metadata by environment

- Cloud
  - Typically netflow data, e.g., AWS, VPC Flow logs
- On-premise
  - Alert metadata: Suricata (most popular), other Network Intrusion Detection (NIDS) tools
  - Flow data: typically collected by a netflow collector
  - Network Security Monitoring (NSM) metadata: typically collects protocol metadata for some or all protocols

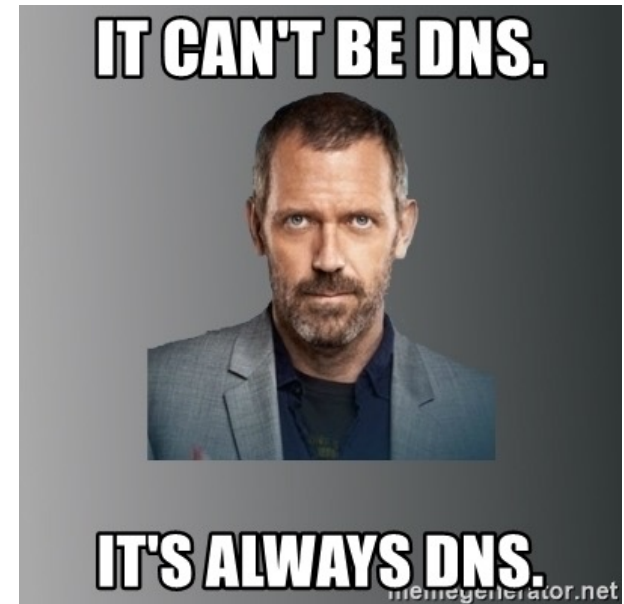
# Core Network Forensics Goals

- Understand the nature of communication.
  - Who is communication.
  - About what.
  - With whom (source / destination pairs).
- Key investigative uses:
  - Bookend an investigation.
  - Confirm key events at network layer.



# Common Protocols - DNS

- Domain Name System – TCP/UDP [53]
  - Always kind of broken.
  - Mapping names to IP addresses.
- DNS Query
  - Given a DNS name -> what IP?
- DNS Response
  - IP <> Domain mapping.
- DNS caching



# Common Protocols - Mail

- Simple Mail Transfer Protocol (SMTP) [TCP 25, 587, 465]
  - “secure” and “non-secure” ports.
  - “Hello and send” protocol
  - Connects to server -> Sends Content
- Internet Message Access Protocol (IMAP) [TCP 143, 993]
  - Folder support
  - Multi-part (large) messages
- Post Office Protocol (POP3) [TCP 110, 995]

# Common Protocols - Web

- Hypertext Transfer Protocol (HTTP) [TCP 80,443]
  - “secure” and “non-secure” ports.
- Request and response protocol
- HTTP Request
  - Given a Uniform Resource Indicator (URI) -> get me the content.
  - Multiple verbs (GET / PUT / POST)
  - GET – variables are in the URI
  - POST – variables and data are a part of the request (commonly not logged)

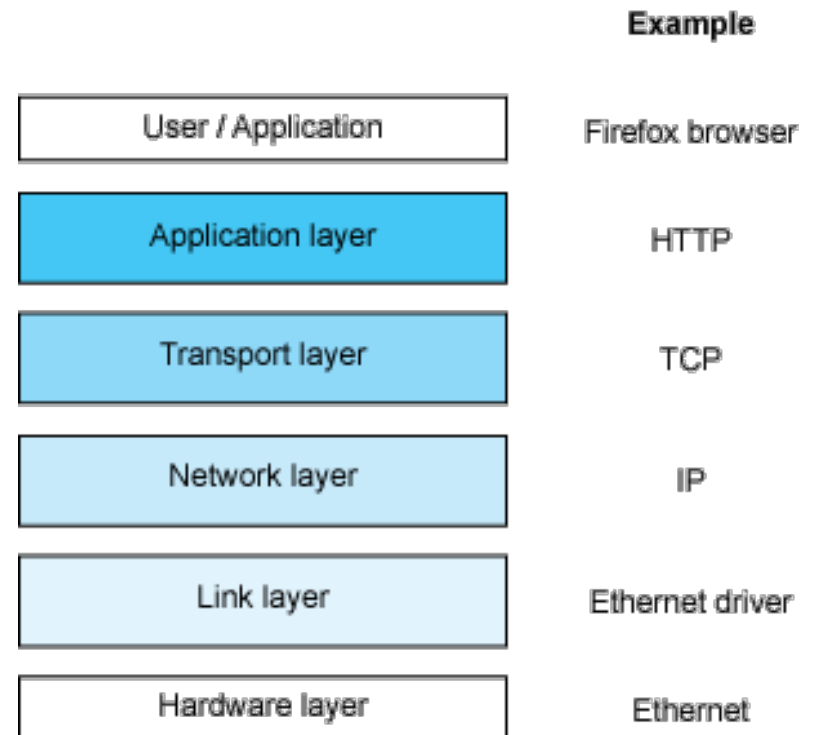
# Common Protocols – Web Cont'd

- Hypertext Transfer Protocol (HTTP) [TCP 80,443]
  - “secure” and “non-secure” ports.
- Request and response protocol
- HTTP Response Codes:
  - 2XX -> Good
  - 3XX -> Redirect
  - 4XX -> Client Error
  - 5XX -> Server Error
- 200 Good
- 404 Not Found
- RFC 2616 // Wikipedia for additional details

# Applied Network Forensics

# Networking Implementation

- Network layers are implemented by different components.
- From network interface drivers -> OS Networking stack -> Application



# Fingerprinting

- Fingerprinting is where we use specific identifying characteristics in evidence (in this case network data) to identify a system or operating system.
- We can use this technique to identify operating systems and (sometimes) installed software on a system.

# Time to Live

- TTL = Time to Live
- IP-layer component
- Packets circulate / have a lifetime
- TTL sets the max lifetime of a packet (IP)



# Fingerprinting Operating Systems

- Windows TTL
  - TTL = 128 (number of router hops)
- Specific update / OS services
  - Windows Update
  - Telemetry  
v10.events.data.microsoft.com
- Auth / Cross-communications
  - DCERPC (445 TCP)
  - EPMAPPER (135 TCP)
- Linux TTL
  - TTL = 64
- Specific update / OS services
  - (distro mirrors)
- Auth / Cross-communications
  - SSH (22 TCP)

# Fingerprinting Applications

- User agents

- Chrome

- dl.google.com

- Firefox

- aus<X>.mozilla.org

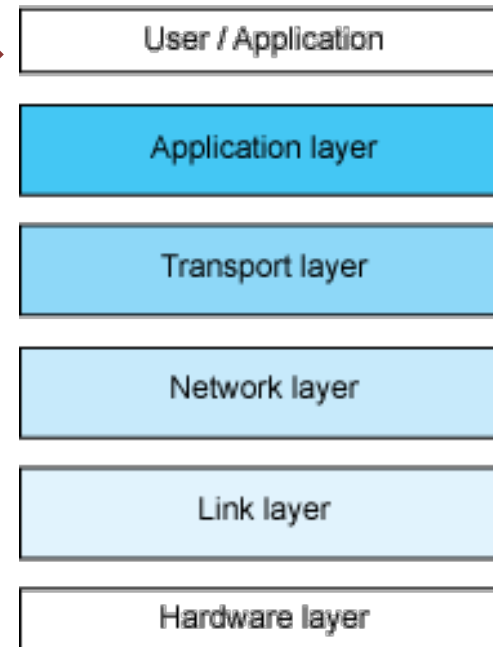
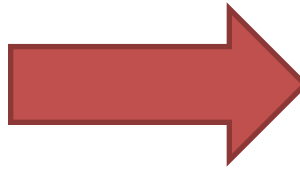
- Productivity

- Office

- nexusrules.officeapps.live.com

- Gsuite

- Googledrive.com



## Example

Firefox browser

HTTP

TCP

IP

Ethernet driver

Ethernet

# Traffic Analysis

- Brim
  - Inputs:
    - Raw Packet Capture (PCAP)
    - Logs (binary or non-binary formats)
      - Binary: ZNG
      - Non-binary: CSV

# Traffic Analysis

- Brim
  - Outputs:
    - Metadata
      - Bro / Zeek
    - Alert data
      - Suricata (Emerging Threats)

# Bro/Zeek

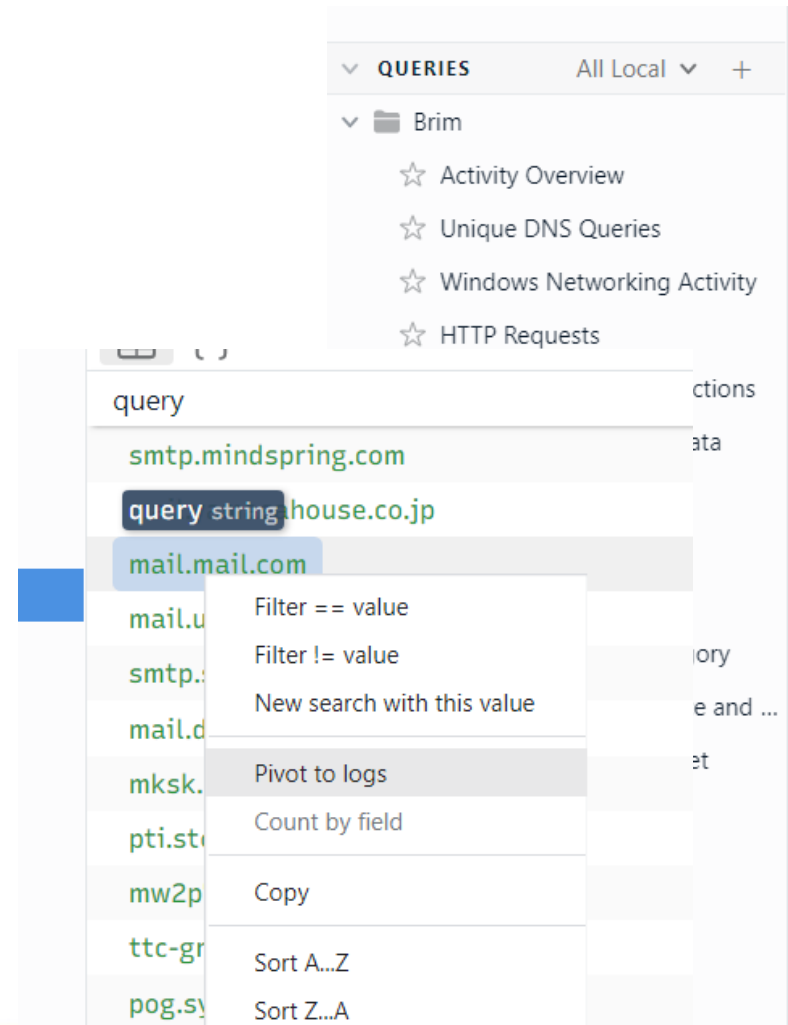


- Used to be called “bro”
- Now called “zeek”
- Takes packet data and produces logs
- Many logs are easy to figure out:
  - Protocol based...
  - http: web traffic
  - conn: netflow
- Some are not
  - “weird”: things Bro thinks are weird
  - “files”: file log generated from traffic (there’s no “file” protocol)



# Working with Brim

- Start with built-in queries
- Then query traffic by protocol:
  - `_path=="dns"`
  - `_path=="http"`
  - ...
- Then summarize using `count()`
- Then drill-down with “pivot to logs”



# Brim Query Language

- “splunk like” uses “|”
- Select > function > present
  - `_path=="dns" | count() by query | sort -r`
  - “count()” = GROUP BY
  - “sort” = sort by amount
- May want to roll / unroll to use “pivot to logs” and other features. Certain aggregations will break features.

This.r.has is not a function [Dismiss](#)

The table view can only render records at the moment.

The table view can only render records at the moment.

# Summarizing Traffic

- Start with built-in bro logs
- Drill into protocols
  - `_path=="dns" | count()` by query
- Drill into alerts
  - `event_type=="alert" | count()`  
by `alert.severity,alert.category |`  
`sort count`

Lab4 - PCAP Evidence.pcap  
204.9 KB 48 MIN

← → `count() by _path | sort -r`

⌵ {...}


_path	count
conn	757
files	607
Error(missing)	402
dns	359
ssl	342
http	268
x509	137
weird	74
smtp	62
dce_rpc	31
notice	15
stats	11
smb_mapping	11
kerberos	9
ntlm	5
capture_loss	4
pe	1
reporter	1

Lab4 - PCAP Evidence

▼ QUERIES AI

▼ Brim

☆ Activity Overview





# Building a Traffic Summary

- Who's talking?
  - Source IP (what is it)
  - Destination IP (what is it)
- What's the nature of the communication?

# Example Traffic Summary

- DNS
  - Clients
  - Local DNS Server
  - Standard / non-standard traffic
    - Queries and responses
    - (ex: A record requests / responses)
  - No non-standard activity observed
    - Look for outliers (query type / etc)

# Example Traffic Summary

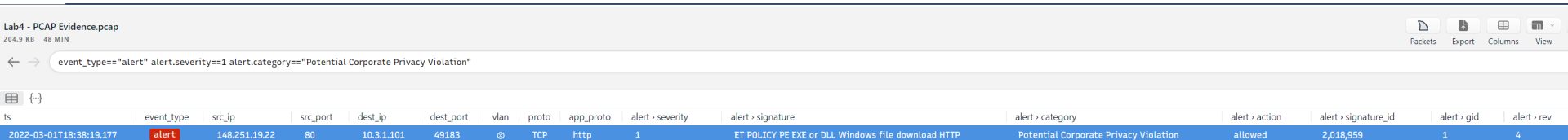
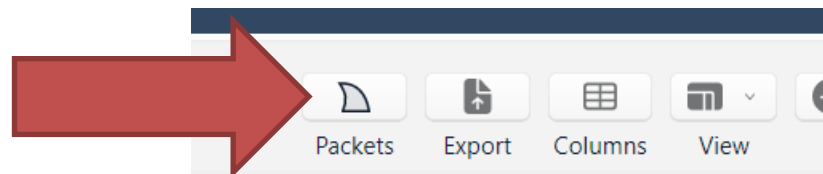
- SMTP
  - Clients (IP / OS / etc)
  - List of email servers
  - Standard / non-standard traffic
    - Ports / protocols / encrypted
    - <normal|not normal> to see # of servers / servers contacted by <workstation|server>
    - Mail client observed
  - No non-standard activity observed
    - Look for outliers / total activity
    - # of servers / etc.

# Is it Bad / Interesting

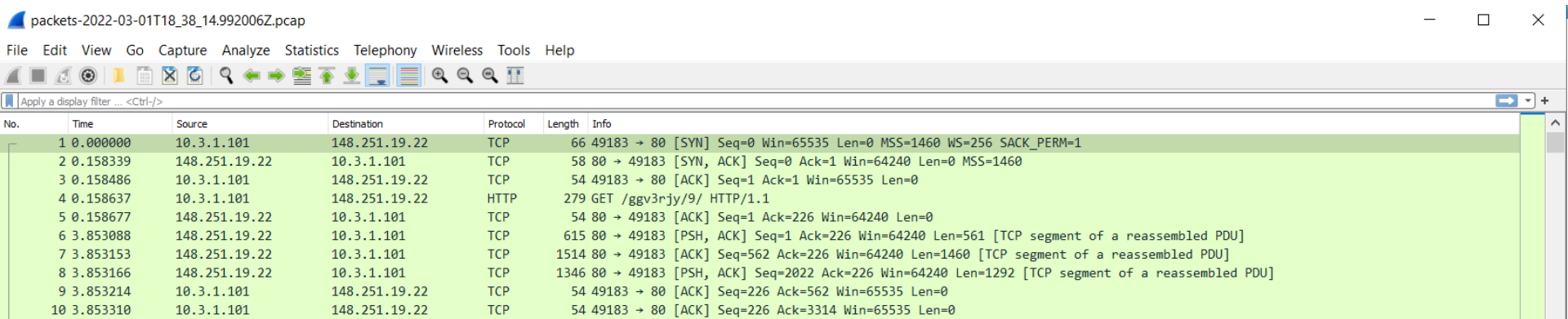
- Lots of approaches – top 3 are:
  - Signature-based matches
    - Typically suricata alerts
  - Outlier analysis
    - Looking for very frequent (beaconing)
    - Looking for very rare (IP / malware download)
  - Key events
    - Binary download

# Working with Wireshark

- Wireshark logo



- This launches wireshark:



# TCP Stream Extraction

- Follow TCP Stream

packets-2022-03-01T18\_38\_14.992006Z.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.3.1.101	148.251.19.22	TCP	66	49183 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK
2	0.158339	148.251.19.22	10.3.1.101	TCP	66	80 → 49183 [ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
3	0.158486	10.3.1.101	148.251.19.22	TCP	66	49183 → 80 [ACK] Seq=1 Ack=1 Win=65535 Len=0
4	0.158637	10.3.1.101	148.251.19.22	TCP	66	49183 → 80 [ACK] Seq=1 Ack=226 Win=64240 Len=0
5	0.158677	148.251.19.22	10.3.1.101	TCP	66	80 → 49183 [ACK] Seq=1 Ack=226 Win=64240 Len=561 [TCP
6	3.853088	148.251.19.22	10.3.1.101	TCP	66	80 → 49183 [ACK] Seq=2022 Ack=226 Win=64240 Len=1292
7	3.853153	148.251.19.22	10.3.1.101	TCP	66	80 → 49183 [ACK] Seq=226 Ack=562 Win=65535 Len=0
8	3.853166	148.251.19.22	10.3.1.101	TCP	66	80 → 49183 [ACK] Seq=226 Ack=3314 Win=65535 Len=0
9	3.853214	10.3.1.101	148.251.19.22	TCP	66	49183 → 80 [ACK] Seq=3314 Ack=226 Win=64240 Len=1460 [TCP
10	3.853310	10.3.1.101	148.251.19.22	TCP	66	49183 → 80 [ACK] Seq=4774 Ack=226 Win=64240 Len=1460 [TCP
11	3.853749	148.251.19.22	10.3.1.101	TCP	66	80 → 49183 [ACK] Seq=6234 Ack=226 Win=64240 Len=1208
12	3.853758	148.251.19.22	10.3.1.101	TCP	66	80 → 49183 [ACK] Seq=7442 Ack=226 Win=64240 Len=1376
13	3.853768	148.251.19.22	10.3.1.101	TCP	66	80 → 49183 [ACK] Seq=7442 Ack=226 Win=64240 Len=1376
14	3.853792	148.251.19.22	10.3.1.101	TCP	66	80 → 49183 [ACK] Seq=7442 Ack=226 Win=64240 Len=1376

Frame 1: 66 bytes on wire (528 bits), 66 bytes captured on interface 0

Ethernet II, Src: HewlettP\_1c:47:ae (00:08:02:1c:47:ae), Dst: 10.3.1.101

Internet Protocol Version 4, Src: 10.3.1.101, Dst: 148.251.19.22

Transmission Control Protocol, Src Port: 49183, Dst Port: 80

Mark/Unmark Packet Ctrl+M

Ignore/Unignore Packet Ctrl+D

Set/Unset Time Reference Ctrl+T

Time Shift... Ctrl+Shift+T

Packet Comments

Edit Resolved Name

Apply as Filter

Prepare as Filter

Conversation Filter

Colorize Conversation

SCTP

Follow

Copy

Protocol Preferences

Decode As...

Show Packet in New Window

TCP Stream Ctrl+Alt+Shift+T

UDP Stream Ctrl+Alt+Shift+U

DCCP Stream Ctrl+Alt+Shift+E

TLS Stream Ctrl+Alt+Shift+S

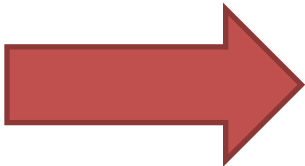
HTTP Stream Ctrl+Alt+Shift+H

HTTP/2 Stream

QUIC Stream

# TCP Stream Extraction

- Follow TCP Stream



```

Wireshark · Follow TCP Stream (tcp.stream eq 0) · packets-2022-03-01T18_38.14.992006Z.pcap
GET /ggv3rjy/9/ HTTP/1.1
Accept: */*
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; Trident/7.0; rv:11.0) like Gecko
Host: diacrestgroup.com
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Tue, 01 Mar 2022 18:38:15 GMT
Server: Apache
X-Powered-By: PHP/5.6.40
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Expires: Tue, 01 Mar 2022 18:38:18 GMT
Content-Disposition: attachment; filename="B7tYH11h5gzY1sx.dll"
Content-Transfer-Encoding: binary
Set-Cookie: 621e681a7dba2=1646159898; expires=Tue, 01-Mar-2022 18:39:18 GMT; Max-Age=60; path=/
Last-Modified: Tue, 01 Mar 2022 18:38:18 GMT
Content-Length: 505856
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/x-msdownload

MZ.....@.....!..L!This program cannot be run in DOS mode.

$.....]/%A.NK..NK..NK.>.&..NK.>.0..NK..NJ.
9LK.....NK.....NK.....NK.....NK.....NK.....NK.....NK.Rich.NK.....PE..L.....)b.....!.....
4.....G.....@.....z.....].....d
.....@.....t.....].@.....text..i~.....
..`rdata.....@..@.data..xg.....p.....@.....rsrc.....@..@.reloc.....

```

# Files in BRIM

- Pulls from Bro/Zeek “file” log
- Gives you a hash



▼ **QUERIES** All Local ▼ +

▼ **Brim**

- ☆ Activity Overview
- ☆ Unique DNS Queries
- ☆ Windows Networking Activity
- ☆ **HTTP Requests**
- ☆ Unique Network Connections
- ☆ Connection Received Data
- ☆ File Activity
- ☆ HTTP Post Requests
- ☆ Show IP Subnets

Lab4 - PCAP Evidence.pcap  
204.9 KB 48 MIN

← → filename!=null | cut \_path, tx\_hosts, rx\_hosts, conn\_uids, mime\_type, filename, md5, sha1

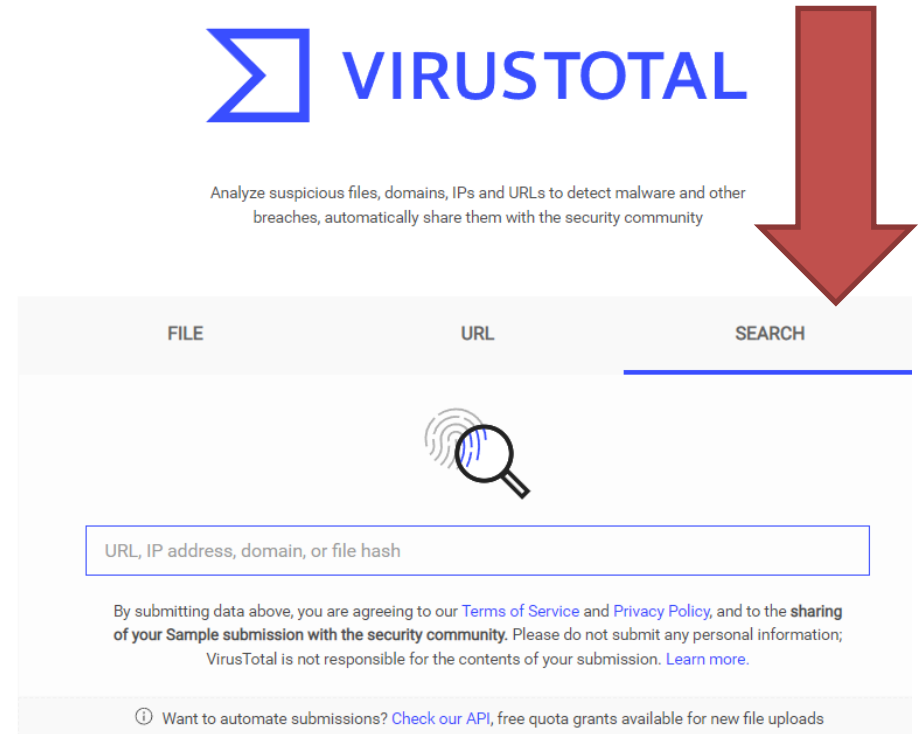


_path	tx_hosts	rx_hosts	conn_uids	mime_type	filename	md5	sha1
files	[[10.3.1.101]]	[[143.90.14.135]]	[[CAv00l3T5V5Sw4d7I]]	application/zip	Invoice # 70376706 A#90739621 4407.zip	5df1c719f5458035f6be2a071ea831db	097ea7a6f7cde395782d4eb86989ac838f772597
files	[[148.251.19.22]]	[[10.3.1.101]]	[[C61RBP1ZpcA95yo275]]	application/x-dosexec	B7tYH11h5gzY1sx.dll	99f59e6f3fa993ba594a3d7077cc884d	839599185ae9b84ef7b4cd6bd274f4785b02fd3f



# Files?

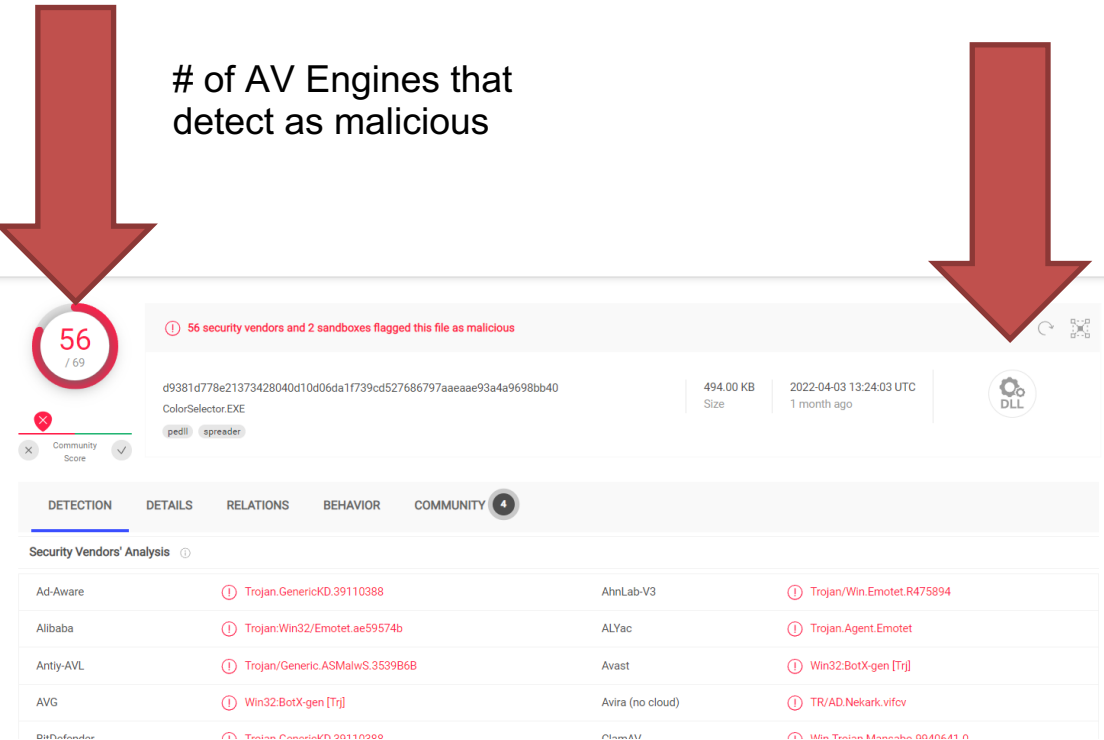
- Sometimes we have full files, sometimes we have metadata.
- In the case of metadata (typically a hash value) – we can go to Virustotal to find out more about the file.



# Virustotal A/V Results

Type of file

# of AV Engines that  
detect as malicious



AV Detection names,  
can help to ID  
malware.

- Among other things VT aggregates AV Engines
- Not always perfect but can be a good barometer / starting point to see if a file is malicious.

# Extracting Files

- PCAP is a binary file format – essentially file extraction can be performed with 2 methods:
  - Bruteforce file headers
  - Look for “file” sections in protocols (mail attachments, etc.)
- NetworkMiner will do this for us, however, there are other options if we’re not getting what we need.

# Using NetworkMiner

- NetworkMiner – Free Edition
  - Enables us to load a PCAP
  - Extract Files
- Start by loading the PCAP
- Then will list and can extract files.

