elastic

Metadata Analysis with Zeek

v 4.0.0

Metadata Analysis with Zeek

1. Zeek Overview
2. Installation & Configuration
3. Displaying and Filtering Logs

Metadata Analysis with Zeek

1. Zeek Overview

2. Installation & Configuration

3. Displaying and Filtering Logs

What is Zeek?

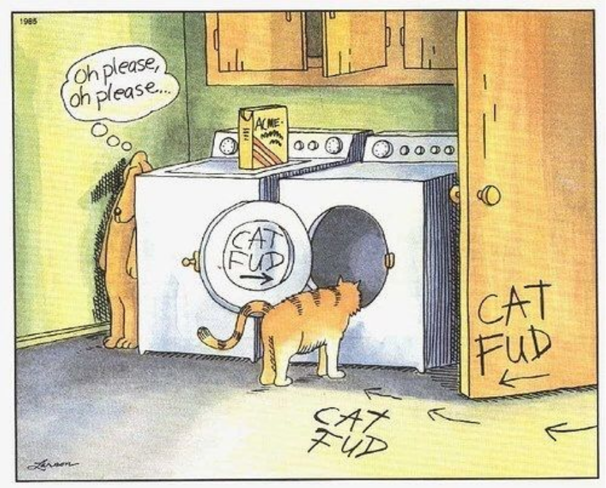
- Passive, open-source traffic analysis platform
- Most often used in cybersecurity, but helpful for performance monitoring and troubleshooting
- Generates rich set of logs that summarize each connection
- Port-independent protocol analysis
- Interactive tutorial available at https://try.zeek.org/

History

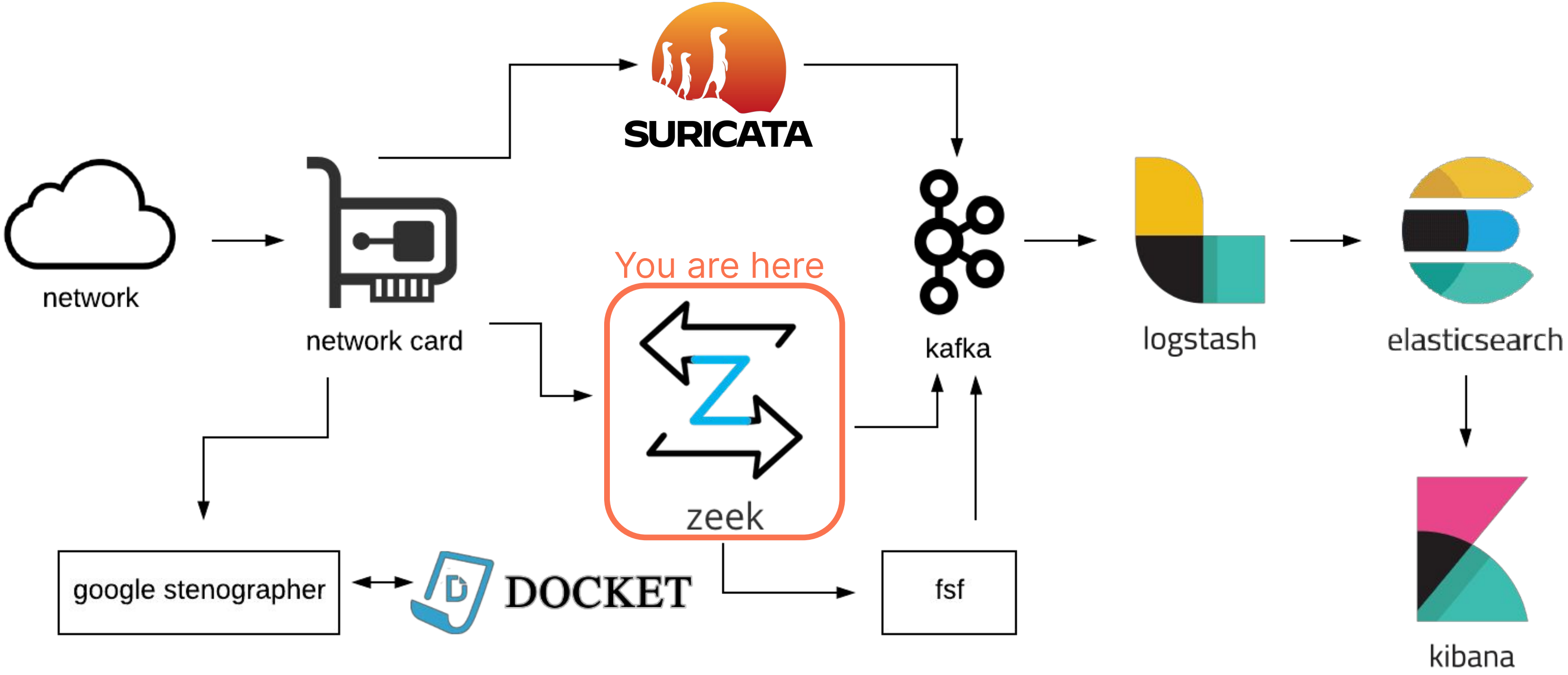
- Created in 1995 by Vern Paxson at Lawrence Berkeley National Laboratory (LBNL) out of research interests

- Previously known as “Bro”, renamed to “Zeek” in 2018

- Heavily used in energy and super-computer research communities since late 90’s



RockNSM


SURICATA
network
network card
You are here
zeek
kafka
logstash
elasticsearch
kibana
google stenographer
DOCKET
fsf

Typical Network log data

Packet captures (PCAP)

SURICATA

Alerts

zeek

Session and Protocol Metadata Logs

Packet Captures (PCAP)

- Pros
  - Allows complete reconstruction of network events
  - Contains all the raw data

- Cons
  - Tremendous amount of storage required
  - Does not scale well

Alerts

- Pros
  - Good for finding “known bad things”
  - Automated alert generation given a rule has been matched

- Cons
  - Prone to false positives and negatives
  - Does not detect events you do not have rules for

Zeek Logs - Session & Protocol Metadata

- Analyzes network data and creates session logs
- Can be used to construct full timeline of events
- See the bigger picture, especially retroactively!
- Port independent protocol analysis
- Uses the terms Originator and Responder
  - originator ≠ source
  - responder ≠ destination

Packet vs Network Events

Packet Events
(Source/Destination)

10.0.0.1 (S)	→	200.1.1.6 (D)
10.0.0.1 (D)	←	200.1.1.6 (S)
10.0.0.1 (S)	→	200.1.1.6 (D)
10.0.0.1 (D)	←	200.1.1.6 (S)

- Based on individual packets
- Source and destination change based on the **direction** of the **communication**

TCPdump
Wireshark
Suricata

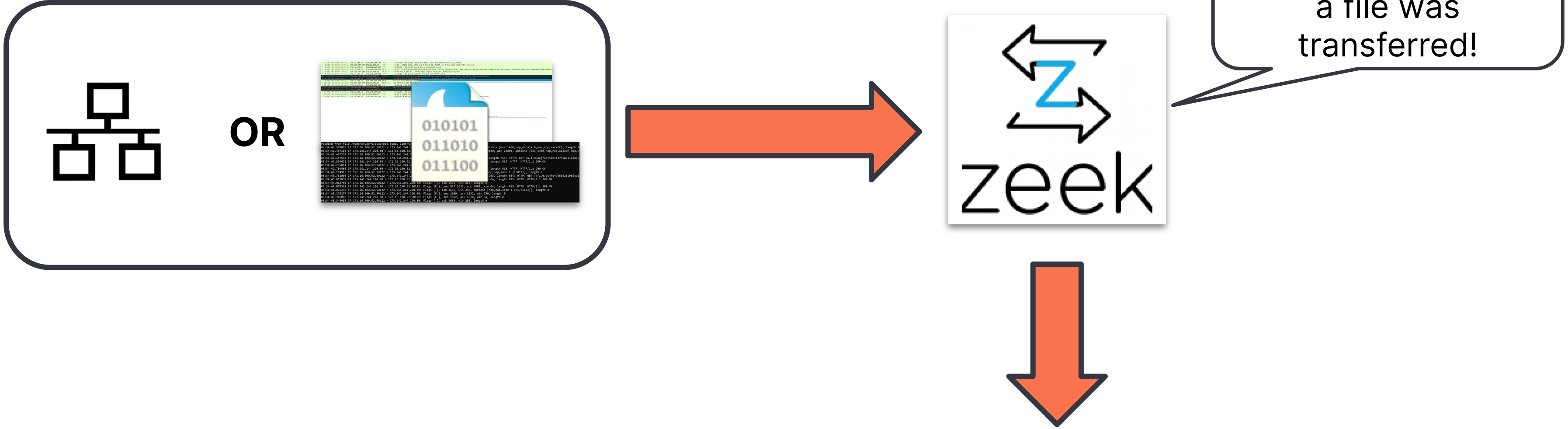
Network Events
(Client/Server)

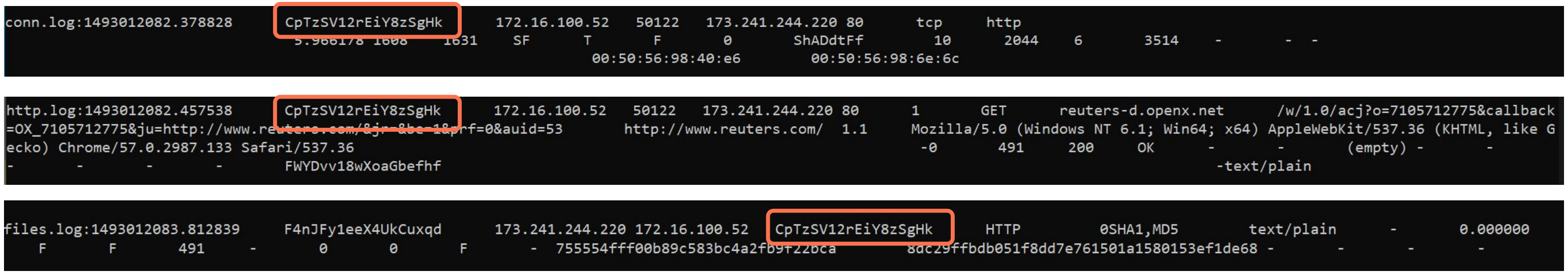
Client (Originator)		**Server (Responder)**
10.0.0.1	→	200.1.1.6
10.0.0.1	←	200.1.1.6
10.0.0.1	→	200.1.1.6
10.0.0.1	←	200.1.1.6

- Based on who **initiated** the conversation
- Client == Originator
- Server == Responder
- Client and Server **do not change** over the course of the connection

Zeek

Zeek Data Flow





conn.log

```
conn.log:1493012082.378828      CpTzSV12rEiY8zSgHk      172.16.100.52   50122   173.241.244.220 80      tcp     http
         5.966178 1608      1631    SF      T       F       0       ShADdtFf        10      2044    6       3514    -       -  -
                                                00:50:56:98:40:e6       00:50:56:98:6e:6c
```

http.log

```
http.log:1493012082.457538      CpTzSV12rEiY8zSgHk      172.16.100.52   50122   173.241.244.220 80      1       GET     reuters-d.openx.net     /w/1.0/acj?o=7105712775&callback
=OX_7105712775&ju=http://www.reuters.com/&jr=&bc=1&prf=0&auid=53        http://www.reuters.com/  1.1     Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like G
ecko) Chrome/57.0.2987.133 Safari/537.36                                                          -0      491     200     OK      -       -       (empty) -       -
-       -       -       -       FWYDvv18wXoaGbefhf                                                                       -text/plain
```

files.log

```
files.log:1493012083.812839     F4nJFy1eeX4UkCuxqd      173.241.244.220 172.16.100.52   CpTzSV12rEiY8zSgHk      HTTP    0SHA1,MD5       text/plain      -       0.000000
   F       F       491     -       0       0       F       -  755554fff00b89c583bc4a2fb9f22bca      8dc29ffbdb051f8dd7e761501a1580153ef1de68 -       -       -       -
```

Zeek Logs

- **Network logs**
- File logs
- netControl
- Detection
- Observations
- Miscellaneous
- Diagnostic

Metadata Analysis with Zeek

1. Zeek Overview

2. **Installation & Configuration**

3. Displaying and Filtering Logs

Zeek Installation

- Zeek can be installed from source, or by using pre-built binary release packages for Linux.
  - Before Zeek can be installed, the required dependencies need to be installed on the target system.
  - Zeek can also make use of optional dependencies, if they are found at build time.
- In our lab environment, we are using the Zeek RPM directly provided by the official Zeek website.
- Can also be run on Docker using the official Docker images.

Zeek Configuration

Path	Description
/usr/bin	● Binaries ● The application that will run
/opt/zeek/etc	● Configurations ● All the settings that manage Zeek preferences
/opt/zeek/logs	● Data ● Default directory where the data logs are stored
/opt/zeek/share/zeek/site	● Scripts ● Custom scripts allowing for customization

/opt/zeek/etc

node.cfg

- Default - standalone
- Used for Zeek clusters

networks.cfg

- Defines the local networks

zeekctl.cfg

- Log Rollover
- Log Path
- Config Directory

Zeek Log Structure

- Defined in the **LogDir** parameter in **/opt/zeek/etc/zeekctl.cfg**

```
/opt/zeek/logs/current

conn.log
dhcp.log
dns.log
http.log
...
```

```
/opt/zeek/logs

2022-04-15
2022-04-16
2022-04-17
2022-04-18
2022-04-19
current -> /opt/zeek/spool
```

Note: Connections are only logged when the connection closes. If a connection is open when the rollover period occurs, it will not be in that log. See corelight/zeek-long-connections for more information.

Metadata Analysis with Zeek

1. Zeek Overview

2. Installation & Configuration

3. **Displaying and Filtering Logs**

Zeek Logs

- Plain ASCII text files
- Column-ized data - `<tab>` delimited
- Module Intent:
  - Need to understand source data
  - Command line tools to filter
- For use with Elasticsearch, logs can be output in json format
  - Final destination:
    - Through Logstash
    - Into Elasticsearch
    - Viewed in Kibana

Zeek CLI - Command Options

- Getting basic help commands

```
zeek --help
```

- Running Zeek

  -C = ignore checksums

  -r = read from a tcpdump or pcap file

```
zeek -Cr [pcap file]
```

Labs 1 & 2
Zeek: Functions Check
Zeek: Additional Configuration

RockNSM Scripts

Rock Bash Aliases

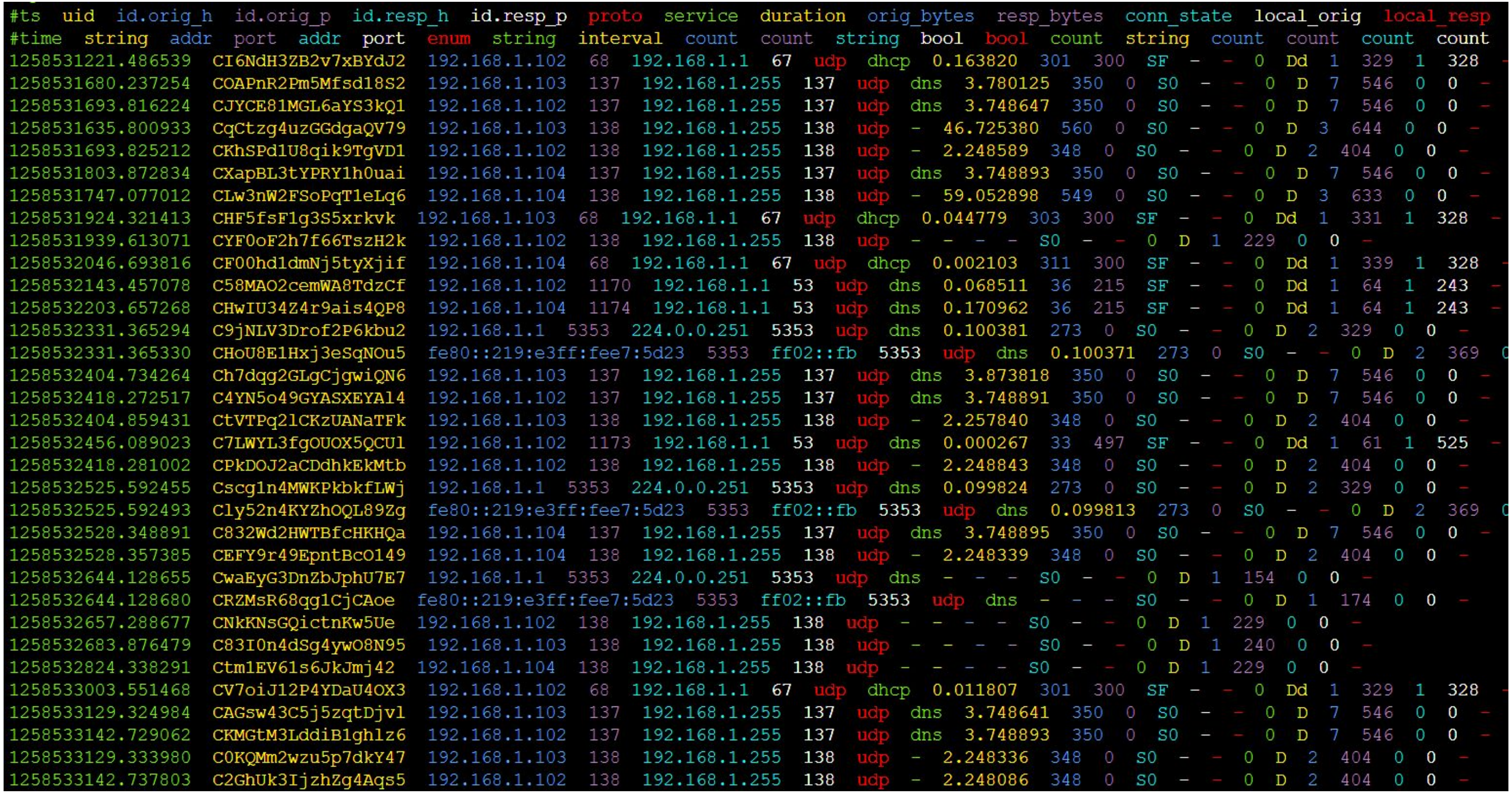
- Available on GitHub in rocknsm/rock
  - /etc/profile.d/rock.sh
  - Included in lab environment
- `lesscolor`
  - performs a `less`, and ***colorizes*** each column

```
lesscolor conn.log
```

- `fields`
  - Lists out each column header

```
fields conn.log
```

lesscolor



```
#ts  uid  id.orig_h  id.orig_p  id.resp_h  id.resp_p  proto  service  duration  orig_bytes  resp_bytes  conn_state  local_orig  local_resp
#time  string  addr  port  addr  port  enum  string  interval  count  count  string  bool  bool  count  string  count  count  count  count
1258531221.486539  CI6NdH3ZB2v7xBYdJ2  192.168.1.102  68  192.168.1.1  67  udp  dhcp  0.163820  301  300  SF  -  -  0  Dd  1  329  1  328  -
1258531680.237254  COAPnR2Pm5Mfsd18S2  192.168.1.103  137  192.168.1.255  137  udp  dns  3.780125  350  0  S0  -  -  0  D  7  546  0  0  -
1258531693.816224  CJYCE81MGL6aYS3kQ1  192.168.1.102  137  192.168.1.255  137  udp  dns  3.748647  350  0  S0  -  -  0  D  7  546  0  0  -
1258531635.800933  CqCtzg4uzGGdgaQV79  192.168.1.103  138  192.168.1.255  138  udp  -  46.725380  560  0  S0  -  -  0  D  3  644  0  0  -
1258531693.825212  CKhSPd1U8qik9TgVD1  192.168.1.102  138  192.168.1.255  138  udp  -  2.248589  348  0  S0  -  -  0  D  2  404  0  0  -
1258531803.872834  CXapBL3tYPRY1h0uai  192.168.1.104  137  192.168.1.255  137  udp  dns  3.748893  350  0  S0  -  -  0  D  7  546  0  0  -
1258531747.077012  CLw3nW2FSoPqT1eLq6  192.168.1.104  138  192.168.1.255  138  udp  -  59.052898  549  0  S0  -  -  0  D  3  633  0  0  -
1258531924.321413  CHF5fsF1g3S5xrkvk  192.168.1.103  68  192.168.1.1  67  udp  dhcp  0.044779  303  300  SF  -  -  0  Dd  1  331  1  328  -
1258531939.613071  CYF0oF2h7f66TszH2k  192.168.1.102  138  192.168.1.255  138  udp  -  -  -  -  S0  -  -  0  D  1  229  0  0  -
1258532046.693816  CF00hd1dmNj5tyXjif  192.168.1.104  68  192.168.1.1  67  udp  dhcp  0.002103  311  300  SF  -  -  0  Dd  1  339  1  328  -
1258532143.457078  C58MAO2cemWA8TdzCf  192.168.1.102  1170  192.168.1.1  53  udp  dns  0.068511  36  215  SF  -  -  0  Dd  1  64  1  243  -
1258532203.657268  CHwIU34Z4r9ais4QP8  192.168.1.104  1174  192.168.1.1  53  udp  dns  0.170962  36  215  SF  -  -  0  Dd  1  64  1  243  -
1258532331.365294  C9jNLV3Drof2P6kbu2  192.168.1.1  5353  224.0.0.251  5353  udp  dns  0.100381  273  0  S0  -  -  0  D  2  329  0  0  -
1258532331.365330  CHoU8E1Hxj3eSqNOu5  fe80::219:e3ff:fee7:5d23  5353  ff02::fb  5353  udp  dns  0.100371  273  0  S0  -  -  0  D  2  369  0
1258532404.734264  Ch7dqg2GLgCjgwiQN6  192.168.1.103  137  192.168.1.255  137  udp  dns  3.873818  350  0  S0  -  -  0  D  7  546  0  0  -
1258532418.272517  C4YN5o49GYASXEYAl4  192.168.1.102  137  192.168.1.255  137  udp  dns  3.748891  350  0  S0  -  -  0  D  7  546  0  0  -
1258532404.859431  CtVTPq2lCKzUANaTFk  192.168.1.103  138  192.168.1.255  138  udp  -  2.257840  348  0  S0  -  -  0  D  2  404  0  0  -
1258532456.089023  C7LWYL3fgOUOX5QCUl  192.168.1.102  1173  192.168.1.1  53  udp  dns  0.000267  33  497  SF  -  -  0  Dd  1  61  1  525  -
1258532418.281002  CPkDOJ2aCDdhkEkMtb  192.168.1.102  138  192.168.1.255  138  udp  -  2.248843  348  0  S0  -  -  0  D  2  404  0  0  -
1258532525.592455  Cscg1n4MWKPkbkfLWj  192.168.1.1  5353  224.0.0.251  5353  udp  dns  0.099824  273  0  S0  -  -  0  D  2  329  0  0  -
1258532525.592493  Cly52n4KYZhOQL89Zg  fe80::219:e3ff:fee7:5d23  5353  ff02::fb  5353  udp  dns  0.099813  273  0  S0  -  -  0  D  2  369  0
1258532528.348891  C832Wd2HWTBfcHKHQa  192.168.1.104  137  192.168.1.255  137  udp  dns  3.748895  350  0  S0  -  -  0  D  7  546  0  0  -
1258532528.357385  CEFY9r49EpntBcO149  192.168.1.104  138  192.168.1.255  138  udp  -  2.248339  348  0  S0  -  -  0  D  2  404  0  0  -
1258532644.128655  CwaEyG3DnZbJphU7E7  192.168.1.1  5353  224.0.0.251  5353  udp  dns  -  -  -  S0  -  -  0  D  1  154  0  0  -
1258532644.128680  CRZMsR68qg1CjCAoe  fe80::219:e3ff:fee7:5d23  5353  ff02::fb  5353  udp  dns  -  -  -  S0  -  -  0  D  1  174  0  0  -
1258532657.288677  CNkKNsGQictnKw5Ue  192.168.1.102  138  192.168.1.255  138  udp  -  -  -  -  S0  -  -  0  D  1  229  0  0  -
1258532683.876479  C83I0n4dSg4ywO8N95  192.168.1.103  138  192.168.1.255  138  udp  -  -  -  -  S0  -  -  0  D  1  240  0  0  -
1258532824.338291  Ctm1EV61s6JkJmj42  192.168.1.104  138  192.168.1.255  138  udp  -  -  -  -  S0  -  -  0  D  1  229  0  0  -
1258533003.551468  CV7oiJ12P4YDaU4OX3  192.168.1.102  68  192.168.1.1  67  udp  dhcp  0.011807  301  300  SF  -  -  0  Dd  1  329  1  328  -
1258533129.324984  CAGsw43C5j5zqtDjvl  192.168.1.103  137  192.168.1.255  137  udp  dns  3.748641  350  0  S0  -  -  0  D  7  546  0  0  -
1258533142.729062  CKMGtM3LddiB1ghlz6  192.168.1.102  137  192.168.1.255  137  udp  dns  3.748893  350  0  S0  -  -  0  D  7  546  0  0  -
1258533129.333980  C0KQMm2wzu5p7dkY47  192.168.1.103  138  192.168.1.255  138  udp  -  2.248336  348  0  S0  -  -  0  D  2  404  0  0  -
1258533142.737803  C2GhUk3IjzhZg4Aqs5  192.168.1.102  138  192.168.1.255  138  udp  -  2.248086  348  0  S0  -  -  0  D  2  404  0  0  -
```

Columns == Fields

```
$ fields conn.log
 1   ts
 2   uid
 3   id.orig_h
 4   id.orig_p
 5   id.resp_h
 6   id.resp_p
 7   proto
 8   service
 ...
```

Linux Command Line Interface

Linux Command Line Interface (CLI)

- Use commands of your choice to view logs!
  - `cat`
  - `less`
  - `head`
- Commands of your choice to manipulate logs!
  - `cut`
  - `awk`
  - `sort`
  - `uniq`
- Useful resource: https://linuxjourney.com/

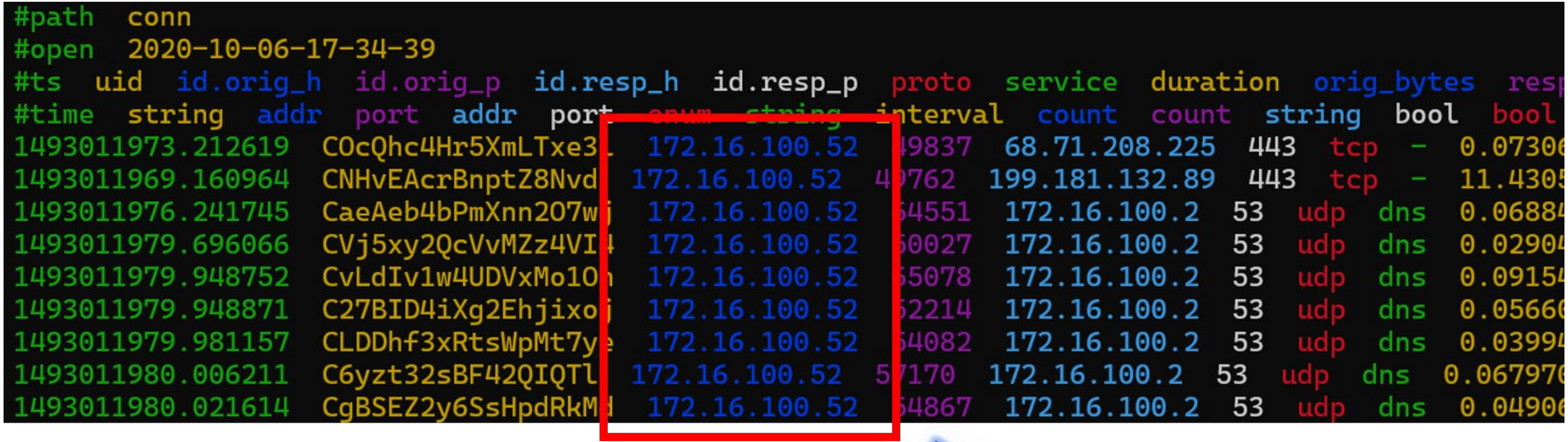
Awk - What is it?

- Scans a file line by line
- Splits each input line into fields
- Compares input line/fields to pattern
- Performs action(s) on matched lines

Syntax:

```
awk options 'selection _criteria {action }' input-file > output-file
```

‘cut’ with ‘awk’



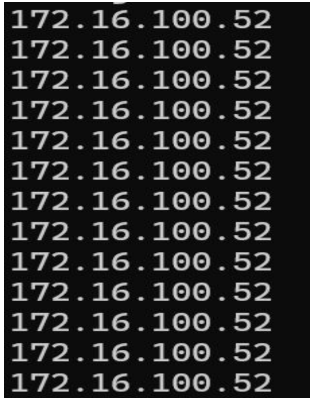
```
#path   conn
#open   2020-10-06-17-34-39
#ts   uid   id.orig_h   id.orig_p   id.resp_h   id.resp_p   proto   service   duration   orig_bytes   resp
#time   string   addr   port   addr   port   enum   string   interval   count   count   string   bool   bool
1493011973.212619   COcQhc4Hr5XmLTxe3   172.16.100.52   9837   68.71.208.225   443   tcp   -   0.07306
1493011969.160964   CNHvEAcrBnptZ8Nvd   172.16.100.52   4 762   199.181.132.89   443   tcp   -   11.4305
1493011976.241745   CaeAeb4bPmXnn207w   172.16.100.52   4551   172.16.100.2   53   udp   dns   0.0688
1493011979.696066   CVj5xy2QcVvMZz4VI   172.16.100.52   0027   172.16.100.2   53   udp   dns   0.0290
1493011979.948752   CvLdIv1w4UDVxMo1O   172.16.100.52   5078   172.16.100.2   53   udp   dns   0.0915
1493011979.948871   C27BID4iXg2Ehjixo   172.16.100.52   2214   172.16.100.2   53   udp   dns   0.0566
1493011979.981157   CLDDhf3xRtsWpMt7y   172.16.100.52   4082   172.16.100.2   53   udp   dns   0.0399
1493011980.006211   C6yzt32sBF42QIQTl   172.16.100.52   5 170   172.16.100.2   53   udp   dns   0.06797
1493011980.021614   CgBSEZ2y6SsHpdRkM   172.16.100.52   4867   172.16.100.2   53   udp   dns   0.0490
```

Using ‘cut’:

```
cut -f 3
```

Using ‘awk’:

```
awk ‘{print $3}’
```



```
172.16.100.52
172.16.100.52
172.16.100.52
172.16.100.52
172.16.100.52
172.16.100.52
172.16.100.52
172.16.100.52
172.16.100.52
172.16.100.52
172.16.100.52
172.16.100.52
172.16.100.52
```

‘grep’ with ‘awk’

Using ‘grep’:

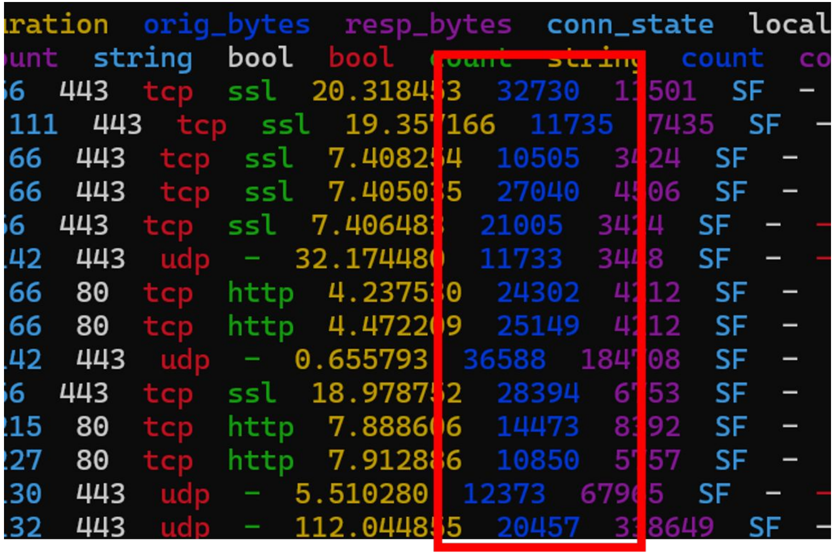
```
grep “172.16.100.52”
```

Using ‘awk’:

```
awk ‘$3 == “172.16.100.52” ’
```

How do you grep for numbers > 10,000?

```
awk ‘$10 >= 10000’
```



```
uration  orig_bytes  resp_bytes  conn_state  local
ount  string  bool  bool  count  string  count  co
6  443  tcp  ssl  20.318453  32730  11501  SF  -
111  443  tcp  ssl  19.357166  11735  7435  SF  -
66  443  tcp  ssl  7.408254  10505  3424  SF  -
66  443  tcp  ssl  7.405035  27040  4506  SF  -
6  443  tcp  ssl  7.406483  21005  3424  SF  -  -
42  443  udp  -  32.174480  11733  3448  SF  -  -
66  80  tcp  http  4.237530  24302  4212  SF  -
66  80  tcp  http  4.472209  25149  4212  SF  -
42  443  udp  -  0.655793  36588  184708  SF  -
6  443  tcp  ssl  18.978752  28394  6753  SF  -
215  80  tcp  http  7.888606  14473  8392  SF  -
227  80  tcp  http  7.912886  10850  5757  SF  -
30  443  udp  -  5.510280  12373  67965  SF  -  -
32  443  udp  -  112.044855  20457  338649  SF  -
```

Sort Sandwich

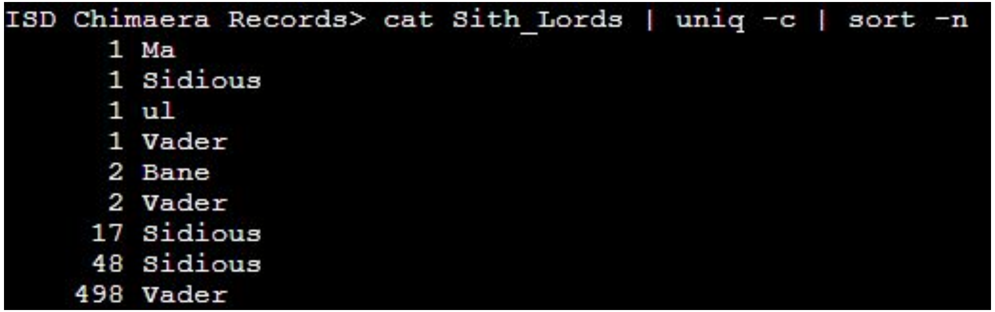
- Allows us to create a simple data table with our logs
- Chain of three commands to sort our data, identify unique instances within our data, and organize our output

```
cat conn.log | cut -f 8 | sort | uniq -c | sort -n
```

Command	**Sort**	**Uniq -c**	**Sort -n**
Why?	Sorts All Like Terms	Shows Unique Values	Sort Results Numerically
Flags	N/A	Count Unique Values	Sort Numerically

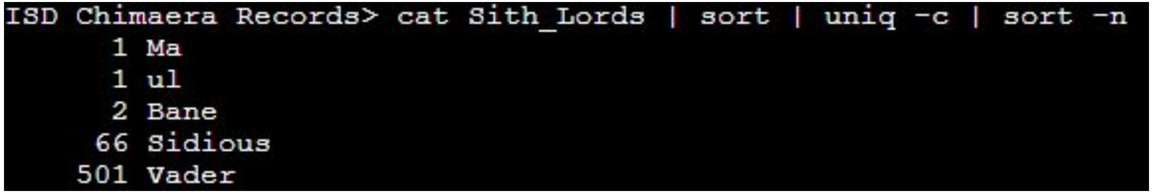
To Sort or not to Sort, that is the question.

- Without the first sort we get duplications within our table, as uniq only checks the line above and below for similarities



```
ISD Chimaera Records> cat Sith_Lords | uniq -c | sort -n
      1 Ma
      1 Sidious
      1 ul
      1 Vader
      2 Bane
      2 Vader
     17 Sidious
     48 Sidious
    498 Vader
```

- When we sort first all of the like strings will be put next to each other, which allows uniq to get an accurate count of our data



```
ISD Chimaera Records> cat Sith_Lords | sort | uniq -c | sort -n
      1 Ma
      1 ul
      2 Bane
     66 Sidious
    501 Vader
```

“Recommended” CLI Process Flow

1. Display the log

```
cat conn.log
```

2. Narrow your focus to the necessary field(s)

```
cat conn.log | cut -f 8
cat conn.log | awk ‘{print $8,$6}’
```

3. “Query” for values within field(s)

```
cat conn.log | cut -f 8 | grep “http”
cat conn.log | awk ‘$8 == “http” && $6 != “80” {print $8,$6}’
```

4. Perform statistics on the data (Sort Sandwich / wc -l) (if needed)

```
cat conn.log | cut -f 8 | grep “http” | wc -l
cat conn.log | awk ‘$8 == “http” && $6 != “80” {print $8,$6}’| sort | uniq -c | sort -n
```

Lab 3: Zeek: Displaying Log Files

Zeek Logs

1. CONN Log
2. Zeek-cut
3. DNS
4. FTP
5. HTTP
6. Kerberos
7. SMB
8. SMTP
9. SSL

Zeek Logs

1. CONN Log
2. Zeek-cut
3. DNS
4. FTP
5. HTTP
6. Kerberos
7. SMB
8. SMTP
9. SSL

CONN - You’ve got the conn(_log)!

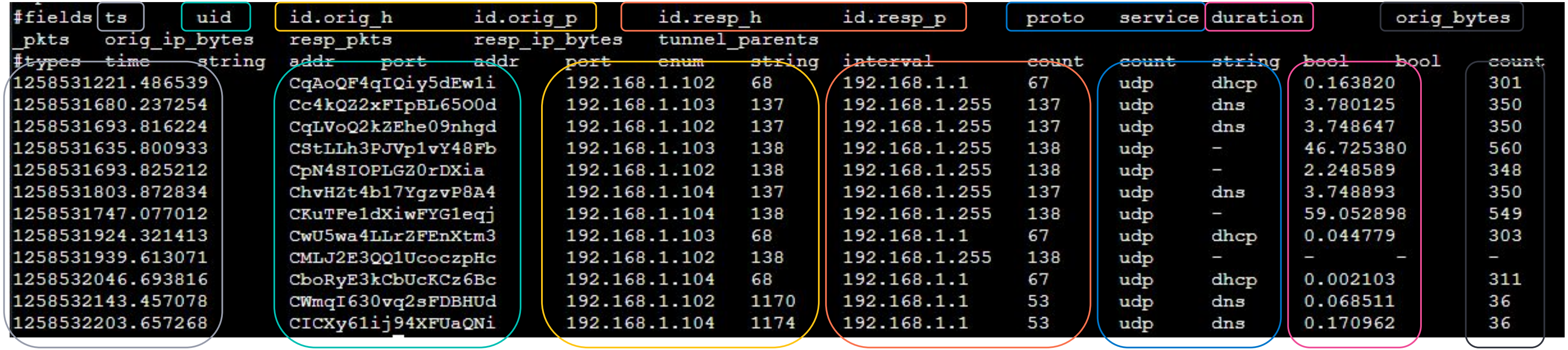
- Zeek logs start with a connection
- Contains NETFLOW and connection metadata
- Connection unique ID is generated in the conn.log
  - Other logs refer to the CONN UID
  - This anchors other events together
- Connections are written to the CONN log when they close
  - This may result in entries in other logs that refer to a connection that is not yet in the conn.log if it has not closed.
- If Zeek sees a connection that it doesn’t know how to categorize, it is still recorded in the conn.log

CONN fields

- Key fields:
  - Timestamp
  - UID
  - originator / responder
  - ports
  - duration
  - number of bytes seen
  - Conn_state
  - Conn_history

Note: Connections are only logged when the connection closes. If a connection is open when the rollover period occurs, it will not be in that log. See corelight/zeek-long-connections for more information.

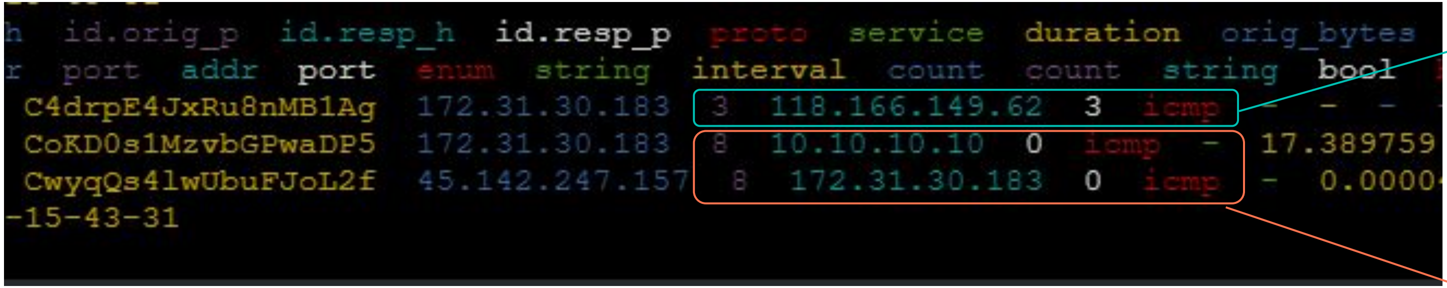
A look at the CONN log



```
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       proto   service duration        orig_bytes
_pkts   orig_ip_bytes   resp_pkts       resp_ip_bytes   tunnel_parents
#types  time    string  addr    port    addr    port    enum    string  interval        count   count   string  bool    bool    count
1258531221.486539       CqAoQF4qIQiy5dEw1i      192.168.1.102   68      192.168.1.1     67      udp     dhcp    0.163820        301
1258531680.237254       Cc4kQZ2xFIpBL65O0d      192.168.1.103   137     192.168.1.255   137     udp     dns     3.780125        350
1258531693.816224       CqLVoQ2kZEhe09nhgd      192.168.1.102   137     192.168.1.255   137     udp     dns     3.748647        350
1258531635.800933       CStLLh3PJVp1vY48Fb      192.168.1.103   138     192.168.1.255   138     udp     -       46.725380       560
1258531693.825212       CpN4SIOPLGZ0rDXia       192.168.1.102   138     192.168.1.255   138     udp     -       2.248589        348
1258531803.872834       ChvHZt4b17YgzvP8A4      192.168.1.104   137     192.168.1.255   137     udp     dns     3.748893        350
1258531747.077012       CKuTFe1dXiwFYG1eqj      192.168.1.104   138     192.168.1.255   138     udp     -       59.052898       549
1258531924.321413       CwU5wa4LLrZFEnXtm3      192.168.1.103   68      192.168.1.1     67      udp     dhcp    0.044779        303
1258531939.613071       CMLJ2E3QQ1UcoczpHc      192.168.1.102   138     192.168.1.255   138     udp     -       -       -       -
1258532046.693816       CboRyE3kCbUcKCz6Bc      192.168.1.104   68      192.168.1.1     67      udp     dhcp    0.002103        311
1258532143.457078       CWmqI630vq2sFDBHUd      192.168.1.102   1170    192.168.1.1     53      udp     dns     0.068511        36
1258532203.657268       CICXy61ij94XFUaQNi      192.168.1.104   1174    192.168.1.1     53      udp     dns     0.170962        36
```

CONN ICMP

- Zeek leverages port fields to log ICMP types/codes



```
h   id.orig_p   id.resp_h   id.resp_p   proto   service   duration   orig_bytes
r   port   addr   port   enum   string   interval   count   count   string   bool
 C4drpE4JxRu8nMB1Ag   172.31.30.183   3   118.166.149.62   3   icmp   -   -   -
 CoKD0s1MzvbGPwaDP5   172.31.30.183   8   10.10.10.10   0   icmp   -   17.389759
 CwyqQs4lwUbuFJoL2f   45.142.247.157   8   172.31.30.183   0   icmp   -   0.0000
-15-43-31
```

Type 3 Code 3
Port
Unreachable

Type 8 and Type 0
Orig: Echo
Request
Resp: Echo Reply

CTF: CONN Log

(pe2.pcap)

Zeek Logs

1. CONN Log
2. Zeek-cut
3. DNS
4. FTP
5. HTTP
6. Kerberos
7. SMB
8. SMTP
9. SSL

Zeek-cut

- Reads ASCII Zeek logs and can display them by column name (instead of column number)
- Field names are separated by spaces
- *zeek-cut -h* to display available flags
  - *zeek-cut -d* will translate timestamps to a human readable format
- Zeek-cut will not read a log file directly
  - you must pipe the log into it

```
cat <log name> | zeek-cut <field name>
```

Zeek-cut examples

```
cat conn.log | zeek-cut proto
```

```
cat conn.log | zeek-cut proto service id.resp_p
```

```
cat conn.log | zeek-cut proto | sort | uniq -c
| sort -rn
```

CTF: ZEEK-CUT

(pe2.pcap)

Zeek Logs

1. CONN Log
2. Zeek-cut
3. **DNS**
4. FTP
5. HTTP
6. Kerberos
7. SMB
8. SMTP
9. SSL

DNS Log - Key Fields

- Ports
  - Zeek includes netbios name service (port 137) in the dns.log
- Protocol
  - tcp vs. udp
- Query
- Qtype
- Rcode
- Answers
- TTLs

CTF: DNS

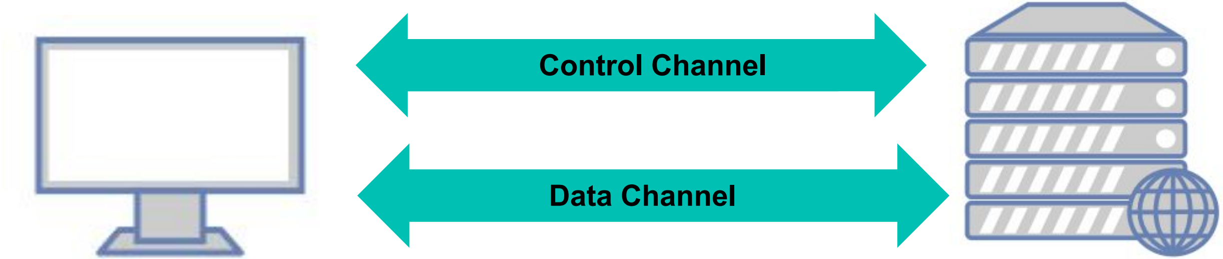
(pe2.pcap)

Zeek Logs

1. CONN Log
2. Zeek-cut
3. DNS
4. **FTP**
5. HTTP
6. Kerberos
7. SMB
8. SMTP
9. SSL

FTP

- Transfer files
- Clear-text sign-in protocol
- Two communication channels
  - control
  - data
- TCP Ports 20/21



Common FTP Client Commands

Command	Description
ascii / binary	Sets the mode of file transfer to [ASCII or binary]
cd (lcd)	Change directory on the remote machine (lcd = local machine)
bye / quit	Exit the FTP environment
open	Open a connection with another computer
close	Terminates a connection with another computer
delete	Delete a file in the current remote directory (same as rm in UNIX)
get (mget)	Copy file from the remote machine to the local machine
put (mput)	Copy file from the local machine to the remote machine

Common FTP Commands

Command	Description
USER	User sends username to FTP server to validation
PASS	User sends password to FTP server to validation
STOR	Tell the server to expect a file transfer
LIST	Lists out file in current directory
PORT (active)	issued by the client to initiate a data connection required to transfer data
PASV	issued by the client to initiate a data connection required to transfer data
EPSV	Command issued by an FTP/S client to signal the server that it wishes to enter into what is known as Extended Passive Mode

Server Return Codes

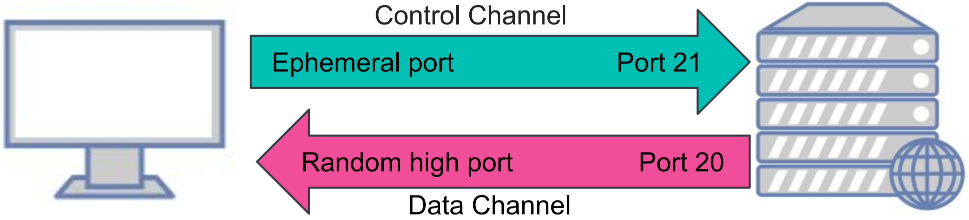
Range	Purpose
1xx	Positive Preliminary Reply
2xx	Positive Completion Reply
3xx	Positive Intermediate Reply
4xx	Transient Negative Completion Reply
5xx	Permanent Negative Completion Reply
6xx	Protected Reply

Range	Purpose
x0x	Syntax
x1x	Information
x2x	Connections
x3x	Authentication and Accounting
x4x	Unspecified (RFC 959)
x5x	File System

Active vs. Passive FTP

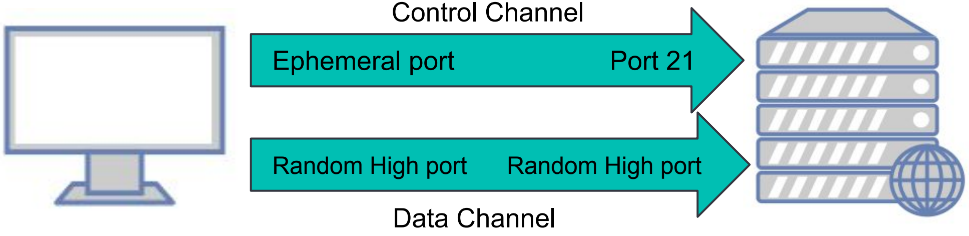
Active Mode

- Client initiates cmd channel
- Server initiates data channel



Passive Mode

- Client initiates cmd channel
- Client initiates data channel



Derivatives

- FTPS (FTP-SSL or FTP Secure)
  - FTP encrypted with SSL/TLS
  - Cmd channel and data channel can be selectively encrypted
- Simple File Transfer Protocol (SFTP)
  - Assigned as “historical” by IETF
  - Complexity between TFTP and FTP
- SSH File Transfer Protocol (SFTP)
  - Inherently encrypts both channels
  - Not simply FTP run over SSH
    - Thus it can’t interoperate with FTP software
  - used by the “secure file transfer program” in Linux
- TFTP
  - simple, lock-step FTP

Attacks/Vulnerabilities

- Anonymous Authentication
  - allows login with user of ‘FTP’ or ‘anonymous’
- Directory Traversal Attack
  - able to create unauthorized files stored outside the root folder
- FTP Bounce Attack
  - discreet port scanning through a proxy
- Cross-Site Scripting (XSS)
  - malicious code sent through browser-side script
- Dridex-based Malware
  - use of ftp sites and creds to avoid detection by email gateways and network policies trusting FTP

FTP Log - Key Fields

- user
- password
- command
- arg
- file_size
- reply_code
- reply_msg

CTF: FTP

(ftp.pcap)

Zeek Logs

1. CONN Log
2. Zeek-cut
3. DNS
4. FTP
5. HTTP
6. Kerberos
7. SMB
8. SMTP
9. SSL

HTTP Log - Key Fields

- method
- host
- uri
- referrer
- user_agent
- status_code
- status_message

CTF: HTTP

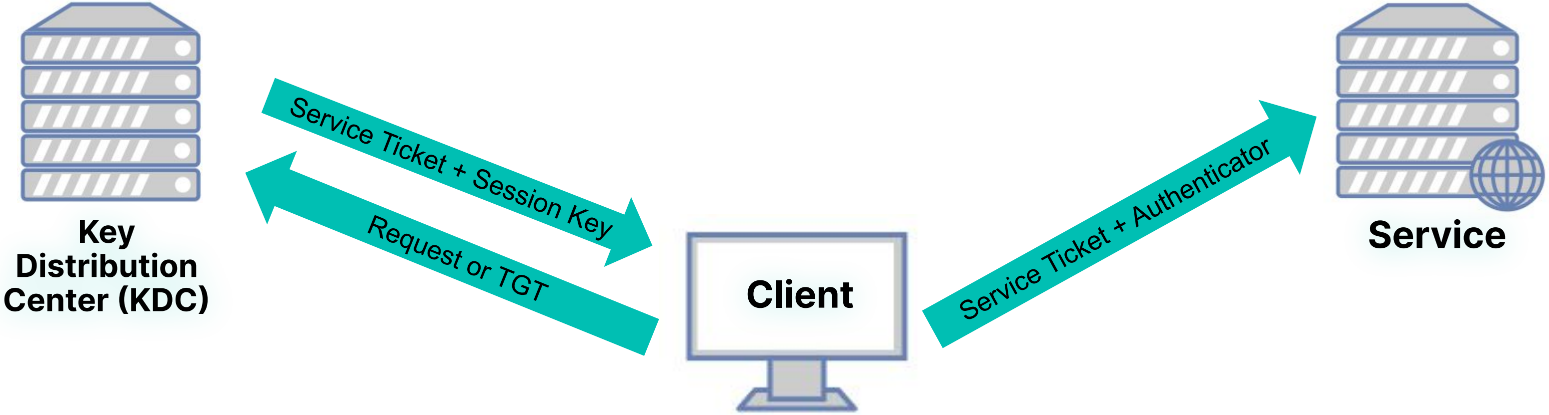
(pe2.pcap)

Zeek Logs

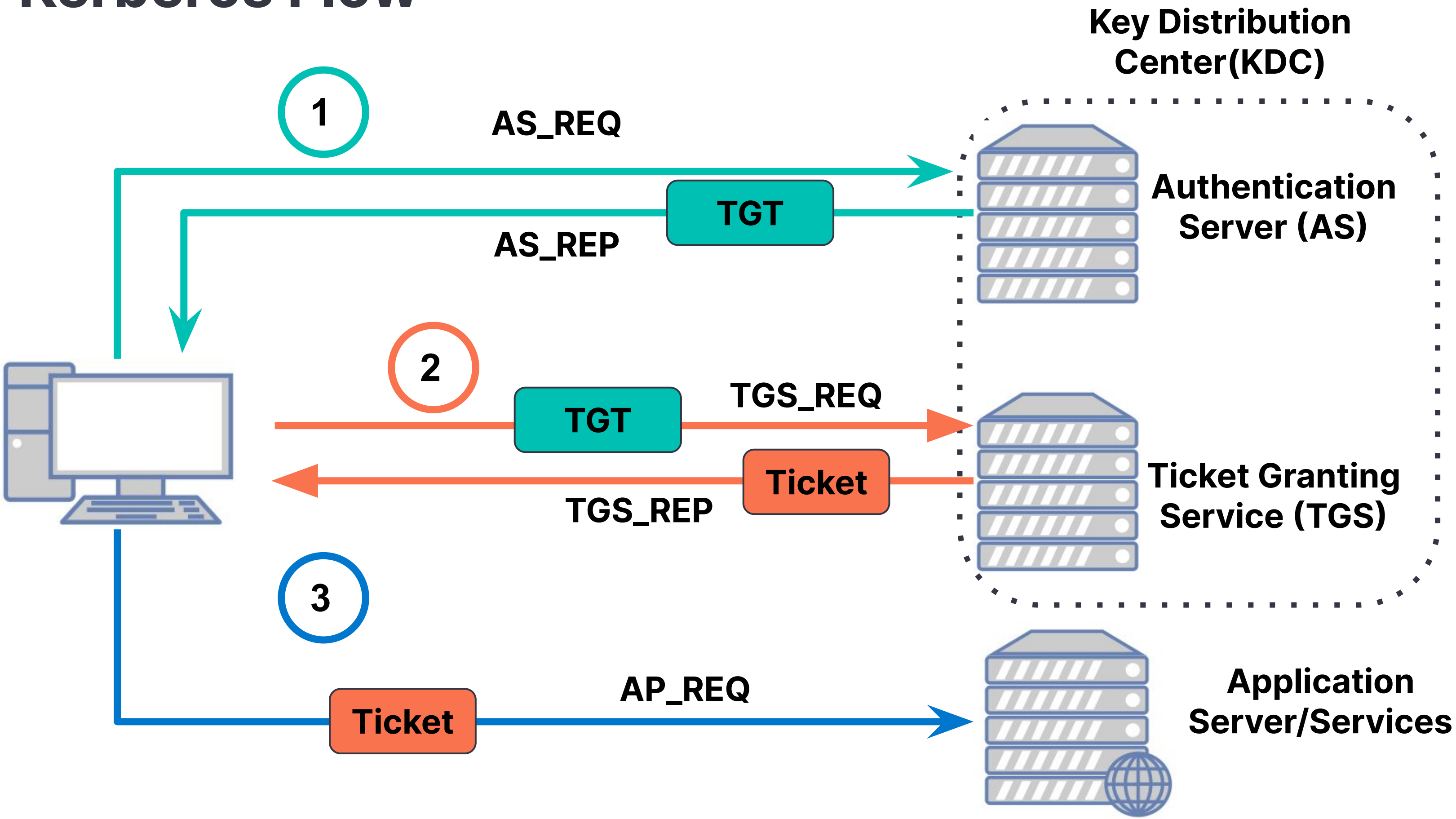
1. CONN Log
2. Zeek-cut
3. DNS
4. FTP
5. HTTP
6. Kerberos
7. SMB
8. SMTP
9. SSL

Kerberos

- Authentication Protocol
- Tickets allowing nodes to talk over a non-secure network
- Default authentication for Windows 2000 and later
- UDP Port 88


Key Distribution Center (KDC)
Service Ticket + Session Key
Request or TGT
Client
Service Ticket + Authenticator
Service

Kerberos Flow

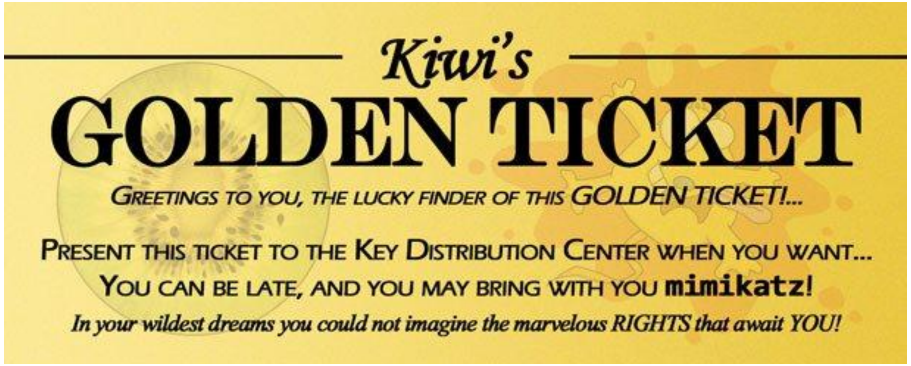

Key Distribution Center(KDC)
1
AS_REQ
TGT
AS_REP
Authentication Server (AS)
2
TGT
TGS_REQ
Ticket
TGS_REP
Ticket Granting Service (TGS)
3
Ticket
AP_REQ
Application Server/Services

Servers and Tickets

- Authentication Service (AS)
  - client authenticates to here
  - issues a Ticket-Granting Ticket (TGT)
    - encrypted using the Ticket-Granting Service’s (TGS) secret key
- Ticket Granting Service (TGS)
  - Client sends the TGT to here when requesting access to a service
  - Issues a ticket for connecting to the requested service
- Service Server (SS)
  - client sends the complete ticket to the SS

Attacks/Vulnerabilities

- Legacy products use DES ciphers instead of AES
  - weak ciphers
- MS14-068 (Kerberos exploit)
  - allowed for elevation of privilege
- Golden Ticket Attack
  - auth token for KRBTGT account



Kiwi's

GOLDEN TICKET

GREETINGS TO YOU, THE LUCKY FINDER OF THIS GOLDEN TICKET!...

PRESENT THIS TICKET TO THE KEY DISTRIBUTION CENTER WHEN YOU WANT...
YOU CAN BE LATE, AND YOU MAY BRING WITH YOU **mimikatz**!
In your wildest dreams you could not imagine the marvelous RIGHTS that await YOU!

Kerberos Log - Key Fields

- request_type
- client
- service
- success
- error_code
- error_msg
- till
- cipher

CTF: Kerberos

(ftp.pcap)

Zeek Logs

1. CONN Log
2. Zeek-cut
3. DNS
4. FTP
5. HTTP
6. Kerberos
7. **SMB**
8. SMTP
9. SSL

SMB

- Enables an app to access resources on a remote server
- TCP Port 139
  - SMB over NetBIOS
- TCP port 445
  - standard after Windows 2000
- Common Internet File System (CIFS)
  - Microsoft dialect (implementation)
  - legacy
  - other dialects include Samba, NQ, and Tuxera

Version Comparison

Version	Release	Description
1 / CIFS	1983 - IBM 1996 - CIFS	- extremely loud, inefficient use of resources - deprecated in June 2013
2.0	Windows Vista Server 2008	- pipelining (high latency) and support for symbolic links - message signing with SHA-256
2.1	Windows 7 Server 2008 R2	- minor performance enhancements - opportunistic locking mechanism
3.0	Windows 8 Server 2012	- Direct Protocol, Multichannel, Transparent Failover - end-to-end encryption and AES based signing
3.0.2	Win 8.1 Server 2012 R2	- allowed for optional disabling of SMB 1*
3.1.1	Windows 10 Server 2016	- supports AES-128; pre-auth checks using SHA-512 - secure negotiations mandatory w/ SMB 2.x and higher

*You can only disable SMBv1 if you don’t have any legacy clients

Attacks/Vulnerabilities

- Sony Pictures hack (2014)
  - Guardians of Peace (GOP), “The Interview”
  - null session attack
- WannaCry Ransomware attack (2017)
  - EternalBlue (leaked by the Shadow Brokers)
- SMBGhost (March 2020)
  - convince a user to connect to malicious SMBv3 server
  - wormlike features
- SMBleed (March 2020)
  - read uninitialized Kernel memory
  - edit compression function

SMB logs generated by Zeek

- Zeek does not generate a single “SMB” log
- Multiple log files may be generated
  - dce_rpc.log
  - kerberos.log
  - ntlm.log
  - smb_cmd.log
  - smb_files.log
  - smb_mapping.log
  - pe.log

SMB_CMD Log - Key Fields

- command, subcommand, argument
  - sent by client
  - subcommand and argument are included, if present
- status
  - from the server
- version
  - 1.0 / 2.0 / 2.1 / 3.0
- tree, tree_service
  - if related, the tree and type of tree
- username

SMB_MAPPING Log - Key Fields

- path
  - name of the tree path
- service
  - type of resource of the tree (disk share, printer share, named pipe, etc.)
- native_file_system
  - the file system of the tree
- share_type
  - for SMB2, the share type will be included
  - for SMB1, the type of share will be deduced and included as well

SMB_FILES Log - Key Fields

- action
  - read / write / open / delete / rename
- path
- name
- size
- prev_name
- times_*
  - modified / accessed / created / changed

CTF: SMB

(smb.pcap)

Zeek Logs

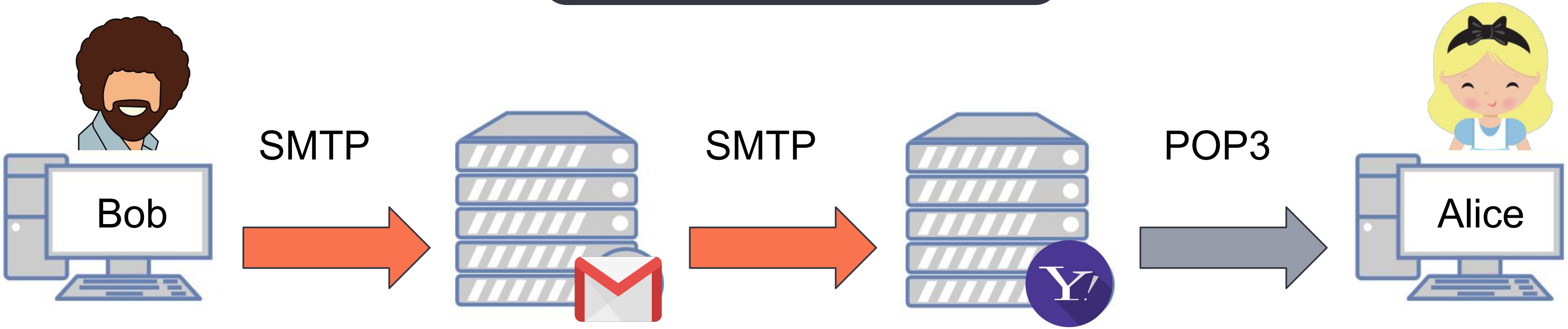
1. CONN Log
2. Zeek-cut
3. DNS
4. FTP
5. HTTP
6. Kerberos
7. SMB
8. **SMTP**
9. SSL

elastic

SMTP

- Store and Forward
- Moves emails on and across networks
- Derived from the Mail Transfer Protocol (MTP)

From: Bob@gmail.com
To: Alice@yahoo.com



SMTP Ports

- Port 25
  - “standard port” (established in 1982)
  - used mostly for SMTP relay (newsletter, spam)
- Port 465
  - originally for SMTPS (SMTP over SSL)
  - has been reassigned and deprecated
- Port 587
  - default port for SMTP submission on modern web
  - supports TLS
- Port 2525
  - popular alternative port for SMTP submission

Message Transport Methods

- Relaying
  - before the days of DNS
  - specifies a sequence of SMTP servers to reach destination
- DNS and Direct Delivery
  - mail exchanger (MX) record
    - email domain name → IP of SMTP domain server
  - used for two transfers:
    - sender’s client to sender’s SMTP server
    - sender’s SMTP server to recipient’s SMTP server

Special Features

- Mail Relaying
  - can be abused for spamming and hacking
- Mail Forwarding
  - for ex-employees that moved to another company
- Mail Gatewaying
  - “translate” TCP/IP email into other email systems
- Address Debugging
  - VRFY command to check validity of an email address
- Mailing List Expansion
  - EXPN command to determine single emails from a mailing list
- “Turning”
  - allows SMTP sender and receiver to change roles

Attacks/Vulnerabilities

- Impersonated SMTP servers
  - converse with a server and manually perform mail transactions
- Account enumeration
  - VRFY command, can script with tons of combinations
- Relay
  - send spam or malware
- Email header disclosures
  - enumerate critical internal information

SMTP Log - Key Fields

- mailfrom
- rcptto
- date
- from / to (these fields can be spoofed!)
- subject
- user_agent
- fuids
  - List of file UID’s seen attached to the message

CTF: SMTP

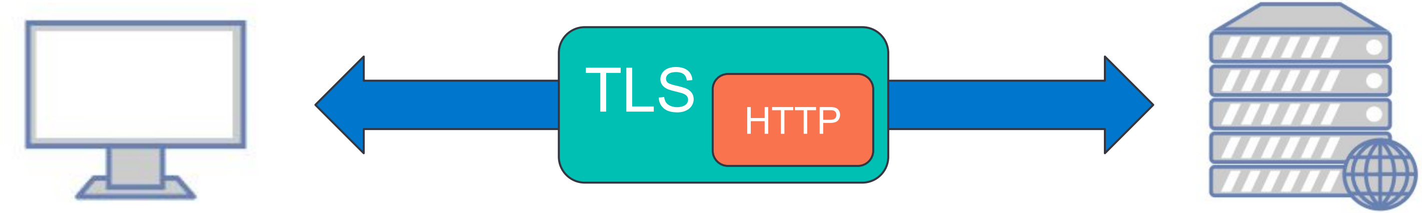
(pcap: smtp.pcap)

Zeek Logs

1. CONN Log
2. Zeek-cut
3. DNS
4. FTP
5. HTTP
6. Kerberos
7. SMB
8. SMTP
9. **SSL**

SSL/TLS Overview

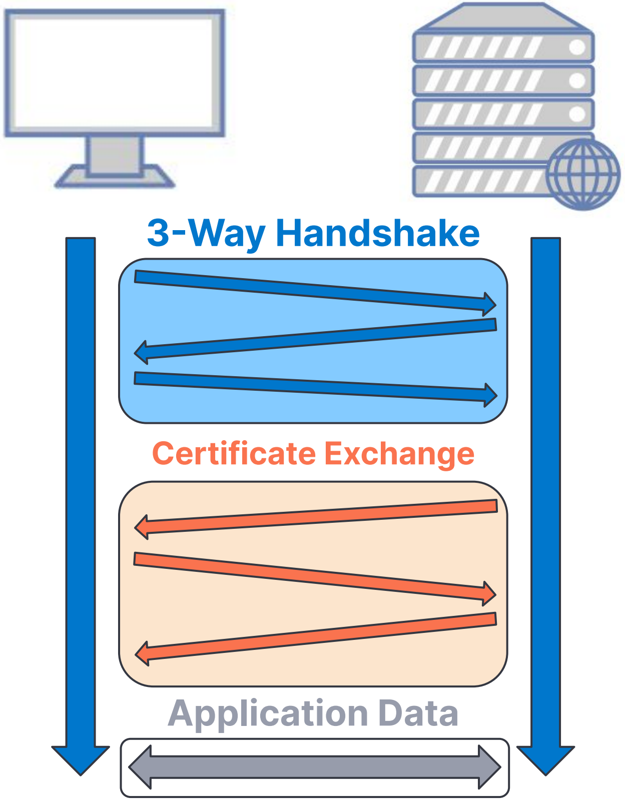
- Extension of HTTP for secure communication (encrypted)
- Port 443 / 8443 (for HTTPS)
  - Port may change depending on the encrypted protocol
- HTTP wrapped in TLS
  - TLS handshake is unencrypted
-



What is TLS and SSL?

- Transport Layer Security (TLS)
  - crypto protocol to encrypt web communications
  - most often for HTTPS communication

- Secure Socket Layer (SSL)
  - deprecated protocol
  - predecessor to TLS

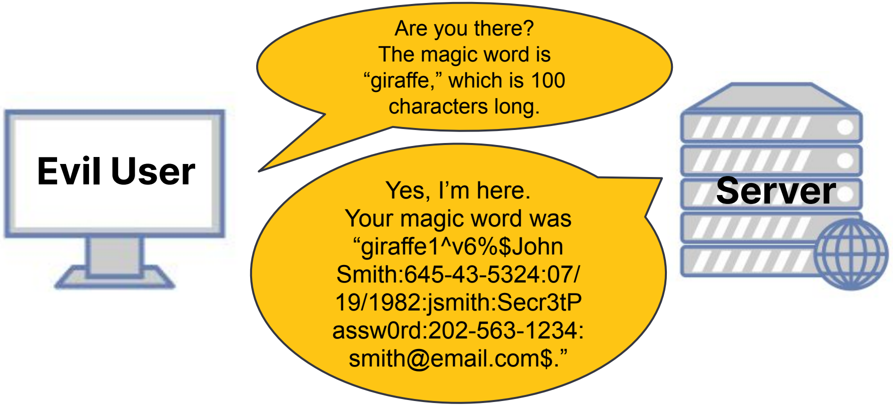


Version Comparison

Version	Released	Deprecated	Description
SSL 1.0	Unpublished	Unpublished	Never went public due to security flaws
SSL 2.0	1995	2011	Lots of security flaws
SSL 3.0	1996	2015	Complete redesign
TLS 1.0	1999	2020	Close to SSL 3.0
TLS 1.1	2006	2020	Improved protection against attacks
TLS 1.2	2008	-	Improved security, support extensions, new cipher suites
TLS 1.3	2018	-	Improved security, drop unsecure features, new cipher suites

Attacks/Vulnerabilities

- SSL Stripping
  - intercepts a redirect and establishes a bridged connection
  - connection is still “secure” from attacker to webpage
- Self-Signed / Wildcard / Expired Certificates
- Heartbleed
  - takes advantage of the “heartbeat” option


Evil User
Are you there? The magic word is “giraffe,” which is 100 characters long.
Yes, I’m here. Your magic word was “giraffe1^v6%$John Smith:645-43-5324:07/19/1982:jsmith:Secr3tPassw0rd:202-563-1234:smith@email.com$.”
Server

SSL Log - Key Fields

- version
- cipher
- curve
- server_name
- subject
- issuer
- validation_status

CTF: SSL

(pe2.pcap)

Quiz

End of Day Quiz - Part 1

1. True or False: Replaying PCAP through Zeek rewrites the timestamps.

2. Which of these is NOT a type of Zeek log?

- Network
- File
- Detection
- Host
- Diagnostic
- Miscellaneous

Quiz - Part 2

3. What happens when Zeek sees a file and the “file extraction” script is enabled?

4. What is the default time interval for Zeek log file rotation (how often a new file is started)?

5. Even when Zeek observes a connection that it doesn’t know how to categorize, it is still recorded in which log?

Quiz - Part 3

6. What command line utility (provided by Zeek) is available to display Zeek logs in a more readable way?

7. Every log will correlate to a unique ID (UID). Which log is this ID generated in?

8. What function does the “-C” flag serve in the following command? “zeek -Cr path/to/pcap”

Quiz - Part 4

9. True or False: Zeek logs are larger and more verbose than a PCAP file.

10. Zeek is a powerful NSM tool. Which of the following best describes what Zeek is?

   a. Message Queue
   b. Tapping Interface
   c. Network Protocol Analyzer