Extending Security Onion

INFO-6081 – Monitoring & Incident Response

FANSHAWE

Learning Outcomes

- Extending SO
- Track Executables with Zeek
- Extracted Executables with Zeek
- The Advanced Persistent Threat Files
- Identifying Downloads of Malicious Binaries
- Proxies
- Checksums
- How Bad Checksums Happen

Extending SO

- While SO is a very powerful tool out of the box, slight system modifications and workflow tasks can add additional features and additional visibility to the setup:
  - Compare MD5 hashes logged with Zeek (Bro) to VirusTotal, or other analysis engines
  - Submit binaries from network traffic to analysis engines
  - Integrate external intelligence from Mandiant’s APT1 report with Zeek to generate alert data

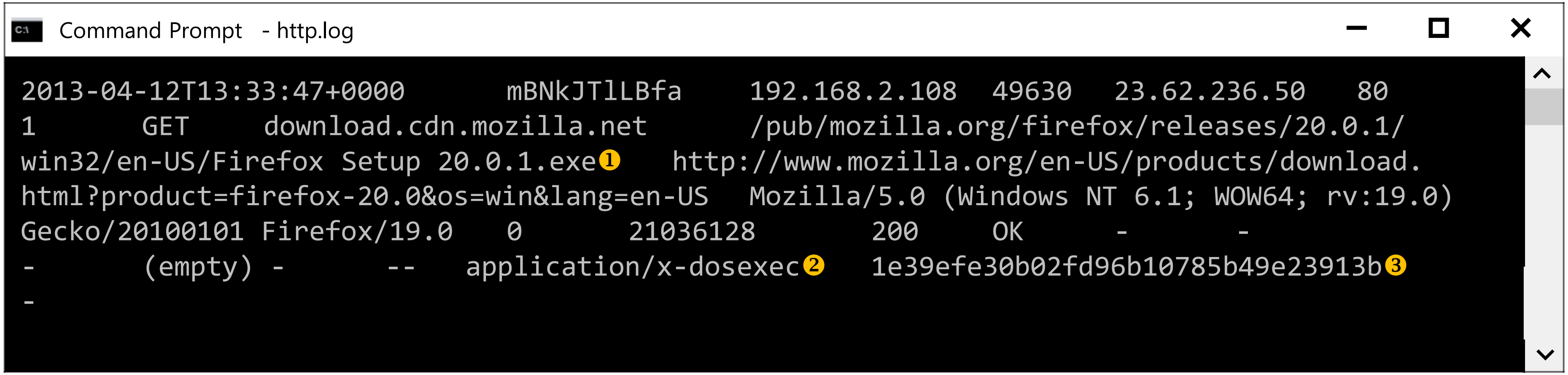
Track Executables with Zeek

- It is a common occurrence that users often need access to new software to complete their work assignments
- While most of this software will come from a reputable source, this is not always the case
- The goals of a phishing campaign is to get access to the organizations systems or data, often by means of tricking a user to download a program from the internet
- Zeek can help you to identify malicious applications by comparing all executables to a known checksum

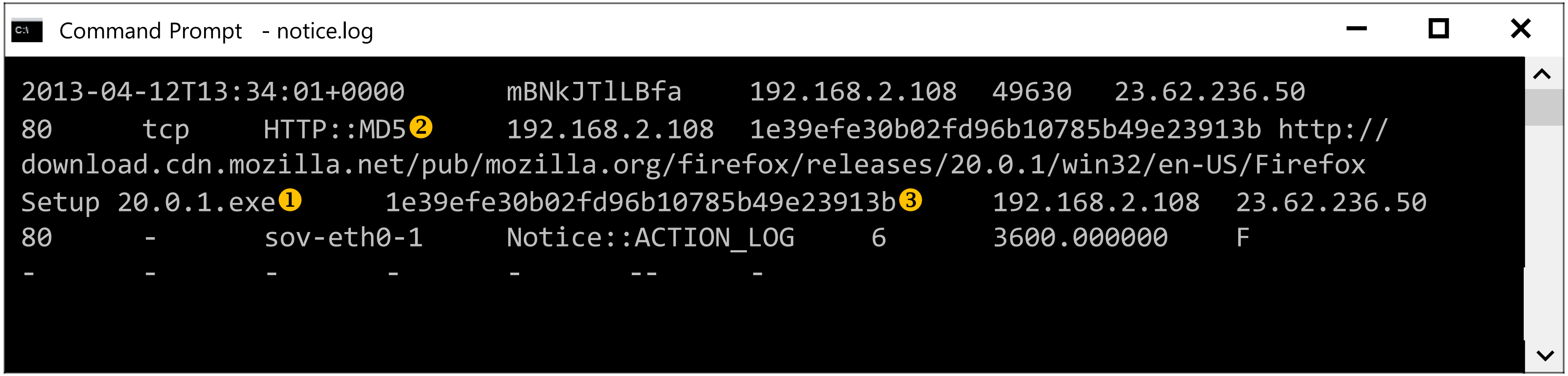
Track Executables with Zeek

- By default, Zeek calculates the MD5 checksum of every executable downloaded by HTTP
- By using hash, we can compare the value for known and unknown files to external databases to identify if a file has been tampered with, without transferring the entire file
- The notice.log file records the values of the MD5 hash

Track Executables with Zeek



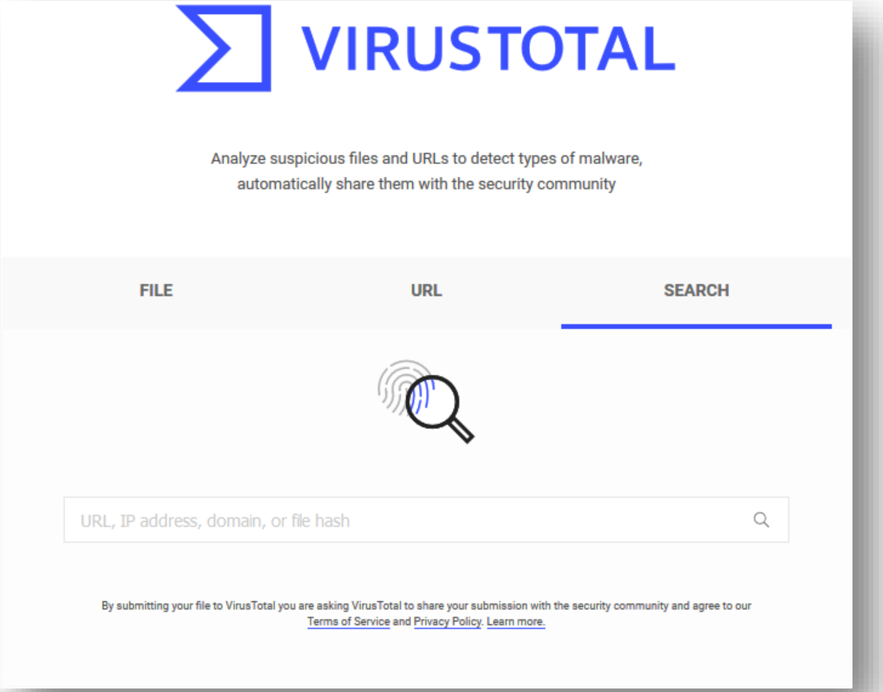
```
2013-04-12T13:33:47+0000      mBNkJTlLBfa    192.168.2.108   49630   23.62.236.50    80
1       GET     download.cdn.mozilla.net      /pub/mozilla.org/firefox/releases/20.0.1/
win32/en-US/Firefox Setup 20.0.1.exe❶    http://www.mozilla.org/en-US/products/download.
html?product=firefox-20.0&os=win&lang=en-US   Mozilla/5.0 (Windows NT 6.1; WOW64; rv:19.0)
Gecko/20100101 Firefox/19.0    0       21036128       200     OK      -       -
-       (empty) -       --    application/x-dosexec❷    1e39efe30b02fd96b10785b49e23913b❸
-
```



```
2013-04-12T13:34:01+0000      mBNkJTlLBfa    192.168.2.108   49630   23.62.236.50
80      tcp     HTTP::MD5❷      192.168.2.108   1e39efe30b02fd96b10785b49e23913b http://
download.cdn.mozilla.net/pub/mozilla.org/firefox/releases/20.0.1/win32/en-US/Firefox
Setup 20.0.1.exe❶      1e39efe30b02fd96b10785b49e23913b❸     192.168.2.108   23.62.236.50
80      -       sov-eth0-1      Notice::ACTION_LOG      6       3600.000000     F
-       -       -       -       -       --      -
```

Track Executables with Zeek

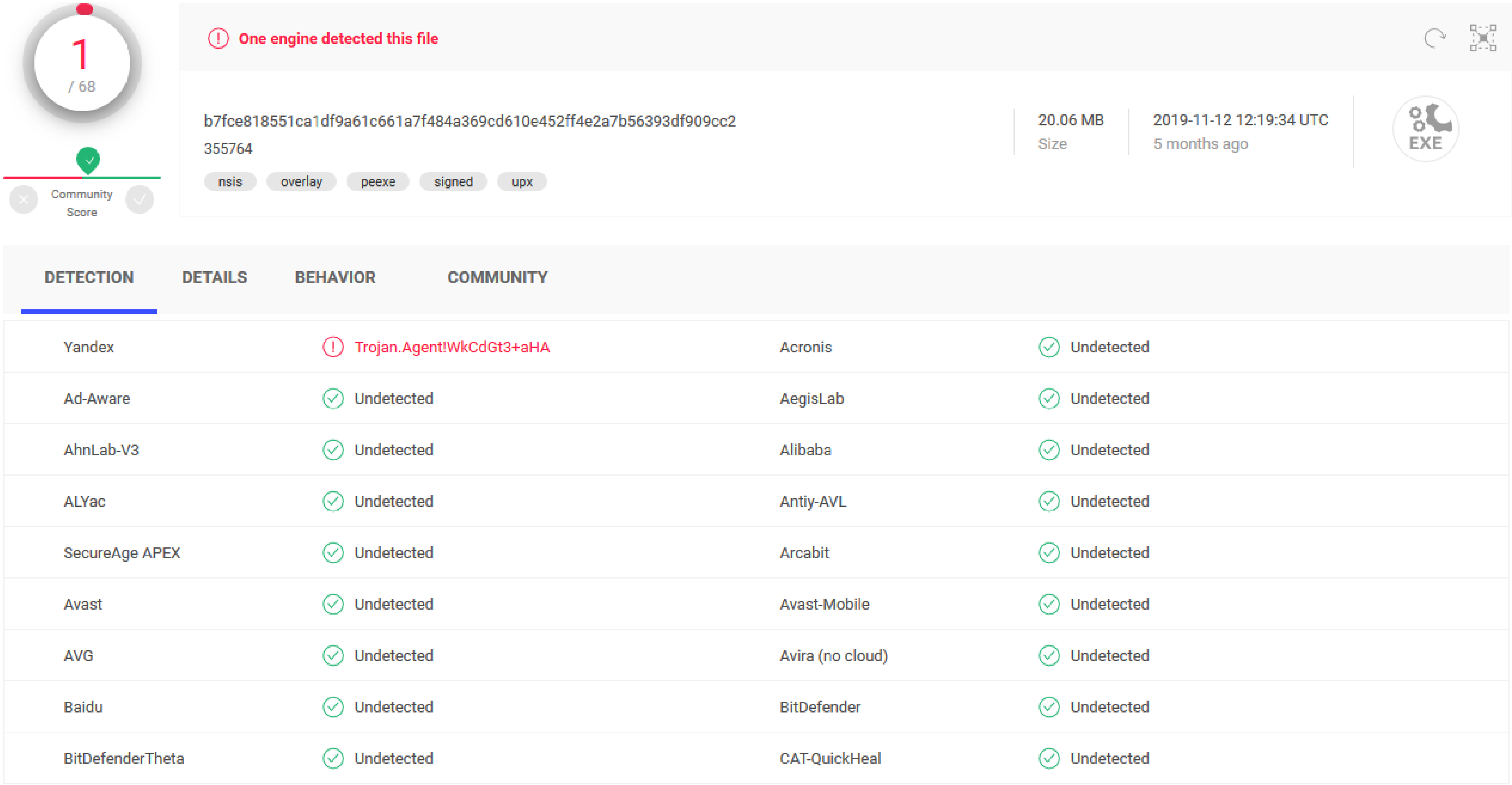
- VirusTotal is one of the most popular resources for additional information regarding binary files
- In addition to submitting a file by upload or URL, you can search for a hash value to see if it exists in the database
- If the hash is recognized, more information is displayed



Analyze suspicious files and URLs to detect types of malware, automatically share them with the security community

By submitting your file to VirusTotal you are asking VirusTotal to share your submission with the security community and agree to our Terms of Service and Privacy Policy. Learn more.

Track Executables with Zeek



1 / 68

Community Score

One engine detected this file

b7fce818551ca1df9a61c661a7f484a369cd610e452ff4e2a7b56393df909cc2
355764

nsis overlay peexe signed upx

20.06 MB
Size

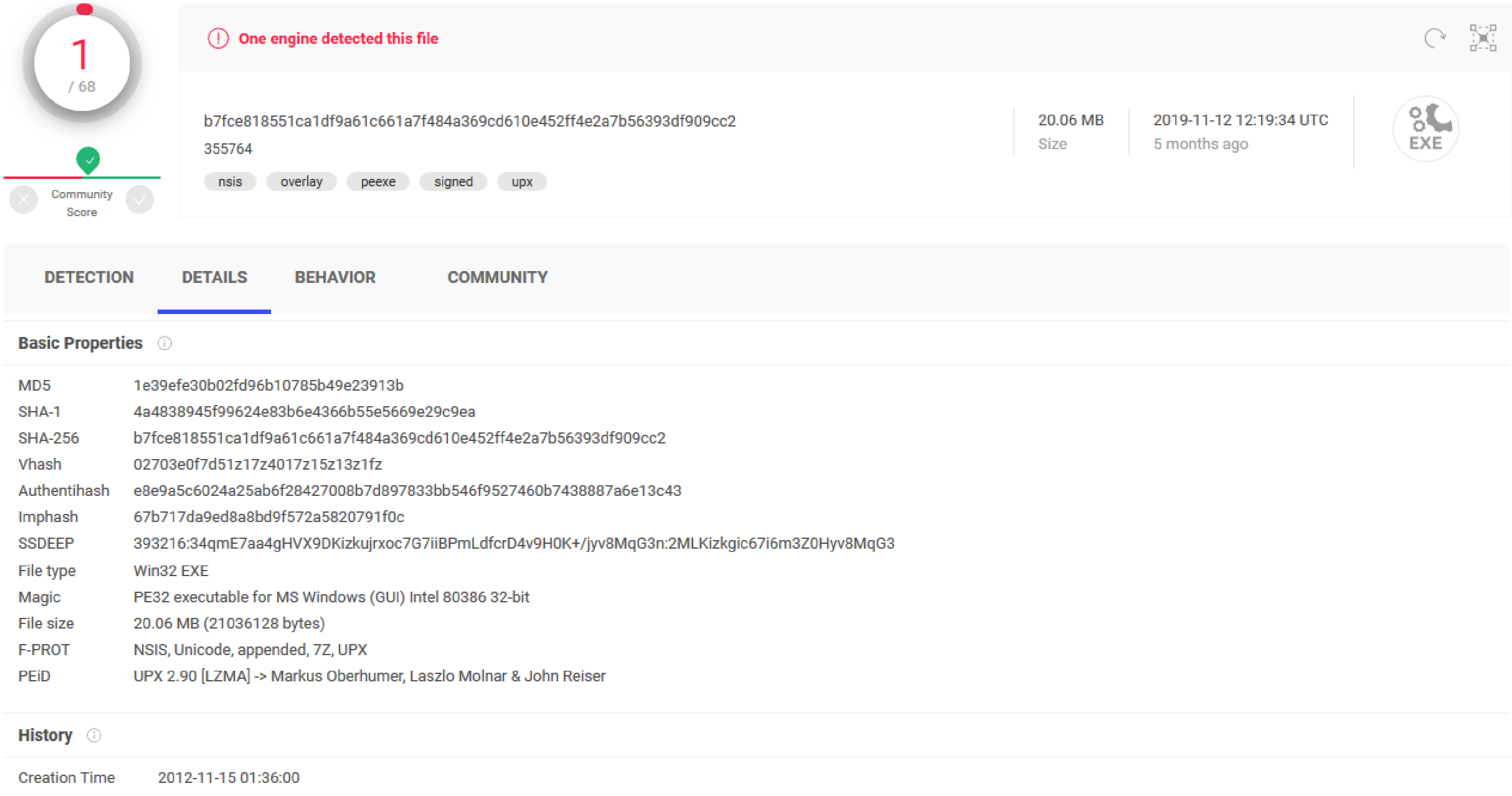
2019-11-12 12:19:34 UTC
5 months ago

EXE

DETECTION DETAILS BEHAVIOR COMMUNITY

Yandex	Trojan.Agent!WkCdGt3+aHA	Acronis	Undetected
Ad-Aware	Undetected	AegisLab	Undetected
AhnLab-V3	Undetected	Alibaba	Undetected
ALYac	Undetected	Antiy-AVL	Undetected
SecureAge APEX	Undetected	Arcabit	Undetected
Avast	Undetected	Avast-Mobile	Undetected
AVG	Undetected	Avira (no cloud)	Undetected
Baidu	Undetected	BitDefender	Undetected
BitDefenderTheta	Undetected	CAT-QuickHeal	Undetected

Track Executables with Zeek



One engine detected this file

b7fce818551ca1df9a61c661a7f484a369cd610e452ff4e2a7b56393df909cc2
355764

20.06 MB
Size

2019-11-12 12:19:34 UTC
5 months ago

Basic Properties

MD5	1e39efe30b02fd96b10785b49e23913b
SHA-1	4a4838945f99624e83b6e4366b55e5669e29c9ea
SHA-256	b7fce818551ca1df9a61c661a7f484a369cd610e452ff4e2a7b56393df909cc2
Vhash	02703e0f7d51z17z4017z15z13z1fz
Authentihash	e8e9a5c6024a25ab6f28427008b7d897833bb546f9527460b7438887a6e13c43
Imphash	67b717da9ed8a8bd9f572a5820791f0c
SSDEEP	393216:34qmE7aa4gHVX9DKizkujrxoc7G7iiBPmLdfcrD4v9H0K+/jyv8MqG3n:2MLKizkgic67i6m3Z0Hyv8MqG3
File type	Win32 EXE
Magic	PE32 executable for MS Windows (GUI) Intel 80386 32-bit
File size	20.06 MB (21036128 bytes)
F-PROT	NSIS, Unicode, appended, 7Z, UPX
PEiD	UPX 2.90 [LZMA] -> Markus Oberhumer, Laszlo Molnar & John Reiser

History

Creation Time	2012-11-15 01:36:00

Extracted Executables with Zeek

- Unlike with older versions of SO, which required modification to do so, Zeek now extracts executable files downloaded over HTTP and FTP to disk automatically
- This makes the process of submitting extracted executables to external databases much easier
- The extracted files can be found in **`/nsm/bro/extracted`**

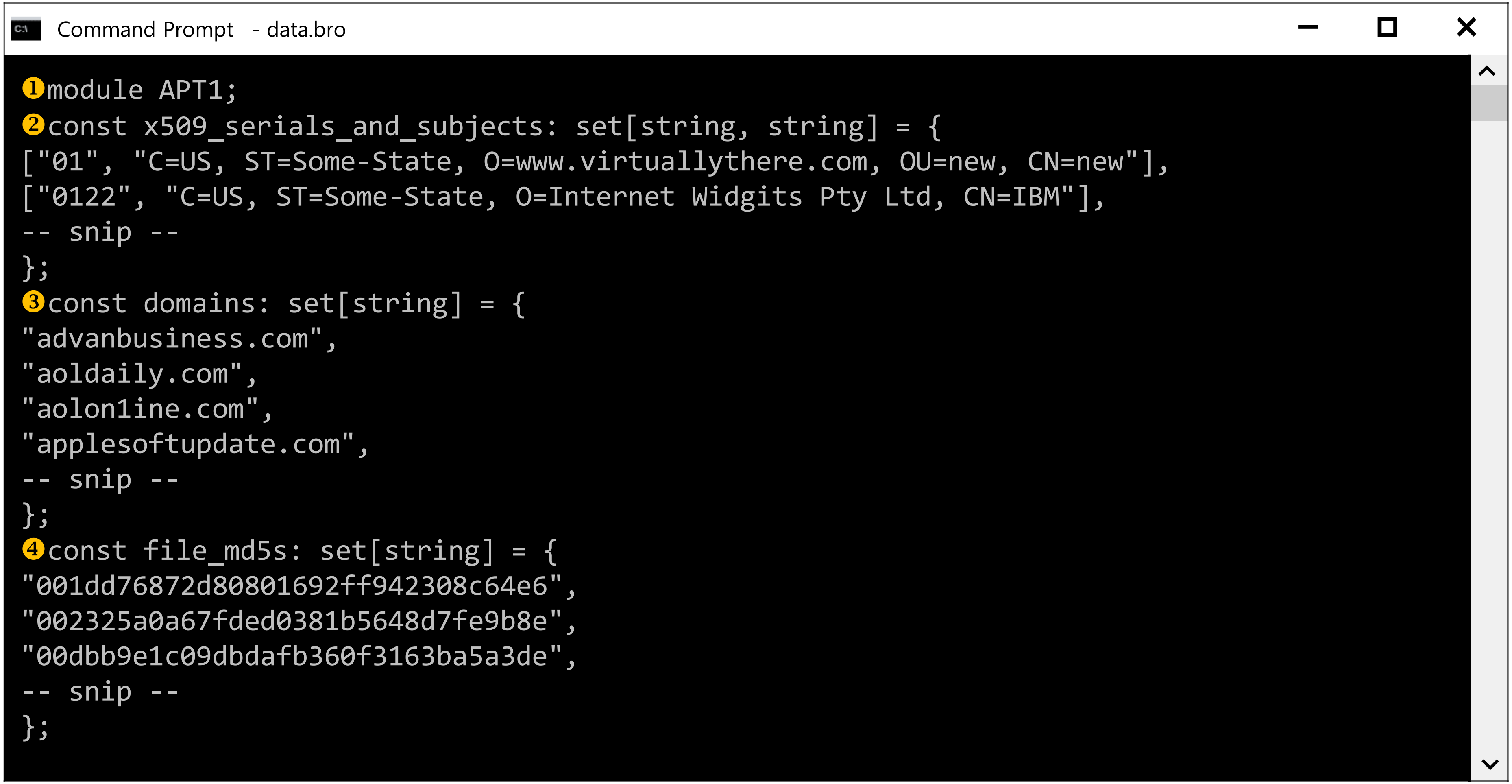
The Advanced Persistent Threat Files - APT1

- In 2013, security consultants Mandiant released a report on a Chinese military unit known as Advanced Persistent Threat 1 (APT1)
- APT1 is the Second Bureau of the Third Department of the General Staff Directorate of the People’s Liberation Army
- Also known by its Military Unit Cover Designator, 61398, this Army team targets organizations within English-speaking countries, to disrupt and steal intellectual property and trade secrets

The Advanced Persistent Threat Files - APT1

- APT1 are known for using the malware: Poision Ivy, Mimikatz, SeaSalt, etc.
- In the report, Mandaint provided 3000 IOCs used by APT1, including Ips, domains, certificate details and malware hashes
- In response, Zeek (then Bro) published a module called APT1, incorporating the information into the scanning engine

The Advanced Persistent Threat Files - APT1

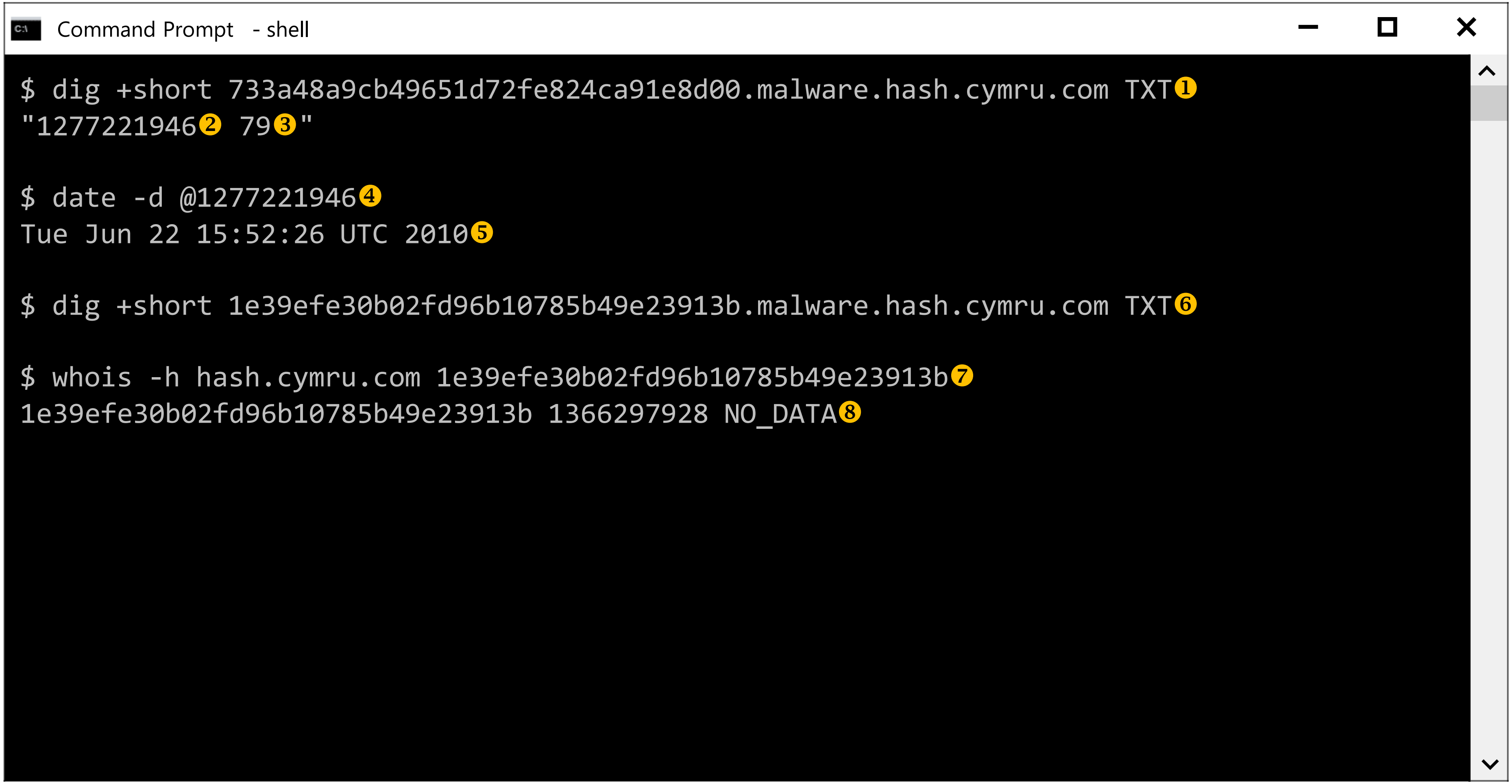


```
❶module APT1;
❷const x509_serials_and_subjects: set[string, string] = {
["01", "C=US, ST=Some-State, O=www.virtuallythere.com, OU=new, CN=new"],
["0122", "C=US, ST=Some-State, O=Internet Widgits Pty Ltd, CN=IBM"],
-- snip --
};
❸const domains: set[string] = {
"advanbusiness.com",
"aoldaily.com",
"aolon1ine.com",
"applesoftupdate.com",
-- snip --
};
❹const file_md5s: set[string] = {
"001dd76872d80801692ff942308c64e6",
"002325a0a67fded0381b5648d7fe9b8e",
"00dbb9e1c09dbdafb360f3163ba5a3de",
-- snip --
};
```

Identifying Downloads of Malicious Binaries

- As discussed earlier, Zeek calculates the MD5 hash of executables downloaded over the network
- SO and Zeek use the Team Cymru Malware Hash Registry to identify files that have been reported as malware
- Team Cymru is an organization dedicated to making the internet more secure through threat intelligence and providing insight for security vendors and network defenders
- The Malware Hash Registry is offered for community use, free of charge

Identifying Downloads of Malicious Binaries

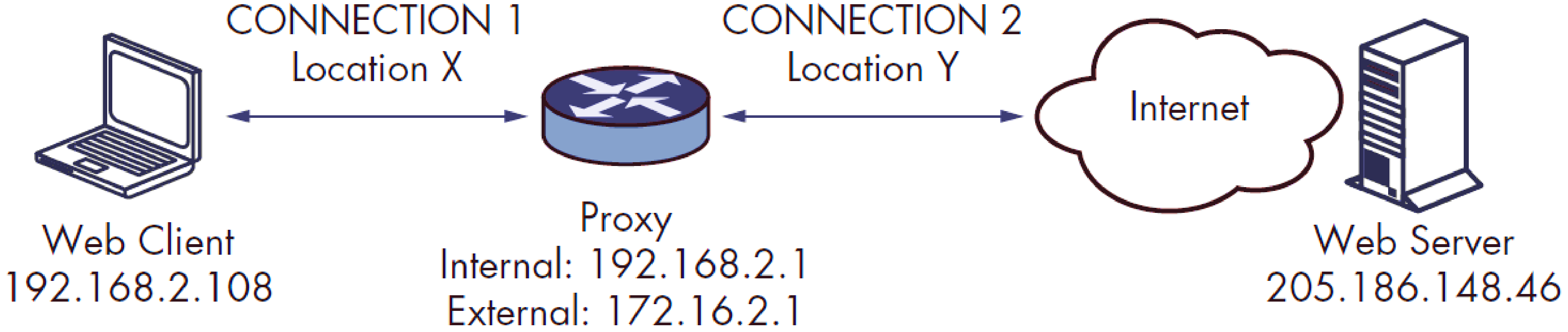


```
$ dig +short 733a48a9cb49651d72fe824ca91e8d00.malware.hash.cymru.com TXT❶
"1277221946❷ 79❸"

$ date -d @1277221946❹
Tue Jun 22 15:52:26 UTC 2010❺

$ dig +short 1e39efe30b02fd96b10785b49e23913b.malware.hash.cymru.com TXT❻

$ whois -h hash.cymru.com 1e39efe30b02fd96b10785b49e23913b❼
1e39efe30b02fd96b10785b49e23913b 1366297928 NO_DATA❽
```

Proxies and Checksums

Proxies

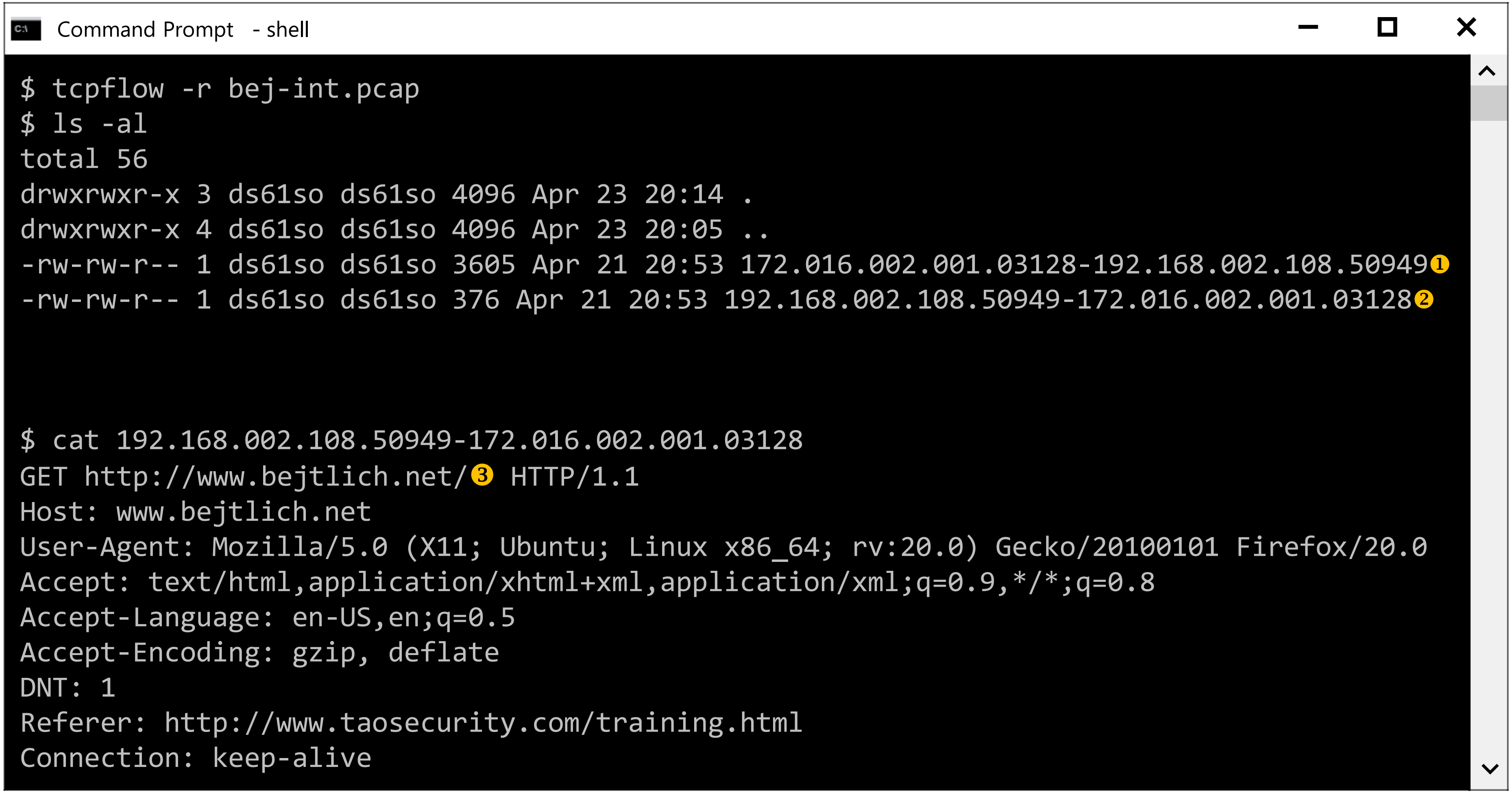
- A web proxy is a piece of network infrastructure used to observe, control and cache HTTP traffic
- Web proxies were traditionally used primarily as a caching device that would accelerate web requests for cached pages
- In today's networks, proxies are primarily focused on filtering and security, as the dynamic content of todays websites does not cache well



Proxies and Visibility

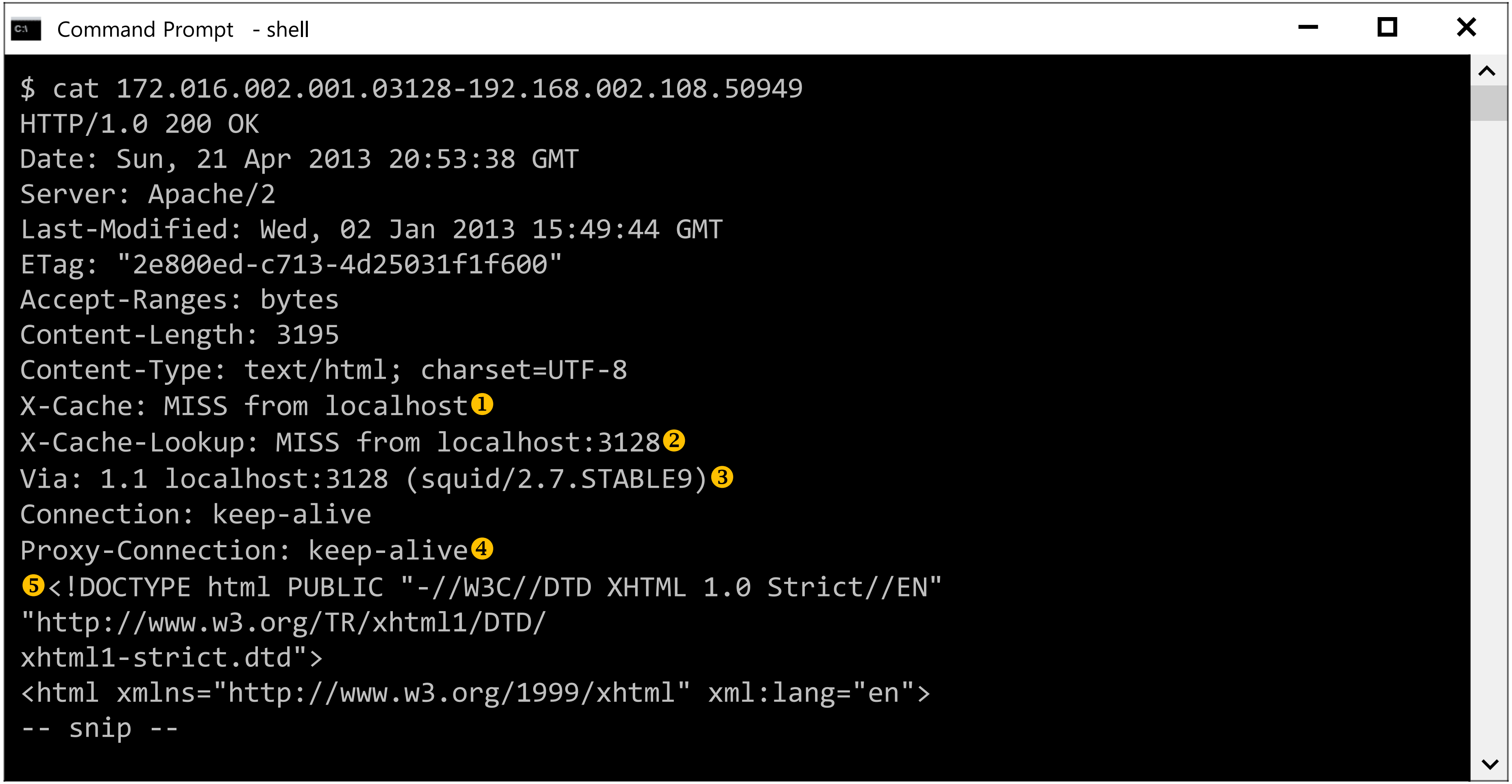
- Proxies limit visibility when conducting NSM
- Instead of seeing source to destination visibility of IP addresses, the client speaks to the proxy, which then creates a new connection to the destination
- The analyst will require access to the proxy logs to gain full visibility of HTTP traffic
- Alternatively, if the proxy allows traffic capture, you can gain additional visibility, but this is not always the case
- The following example is based on captures from a proxy server

Proxy Example – Client to Proxy



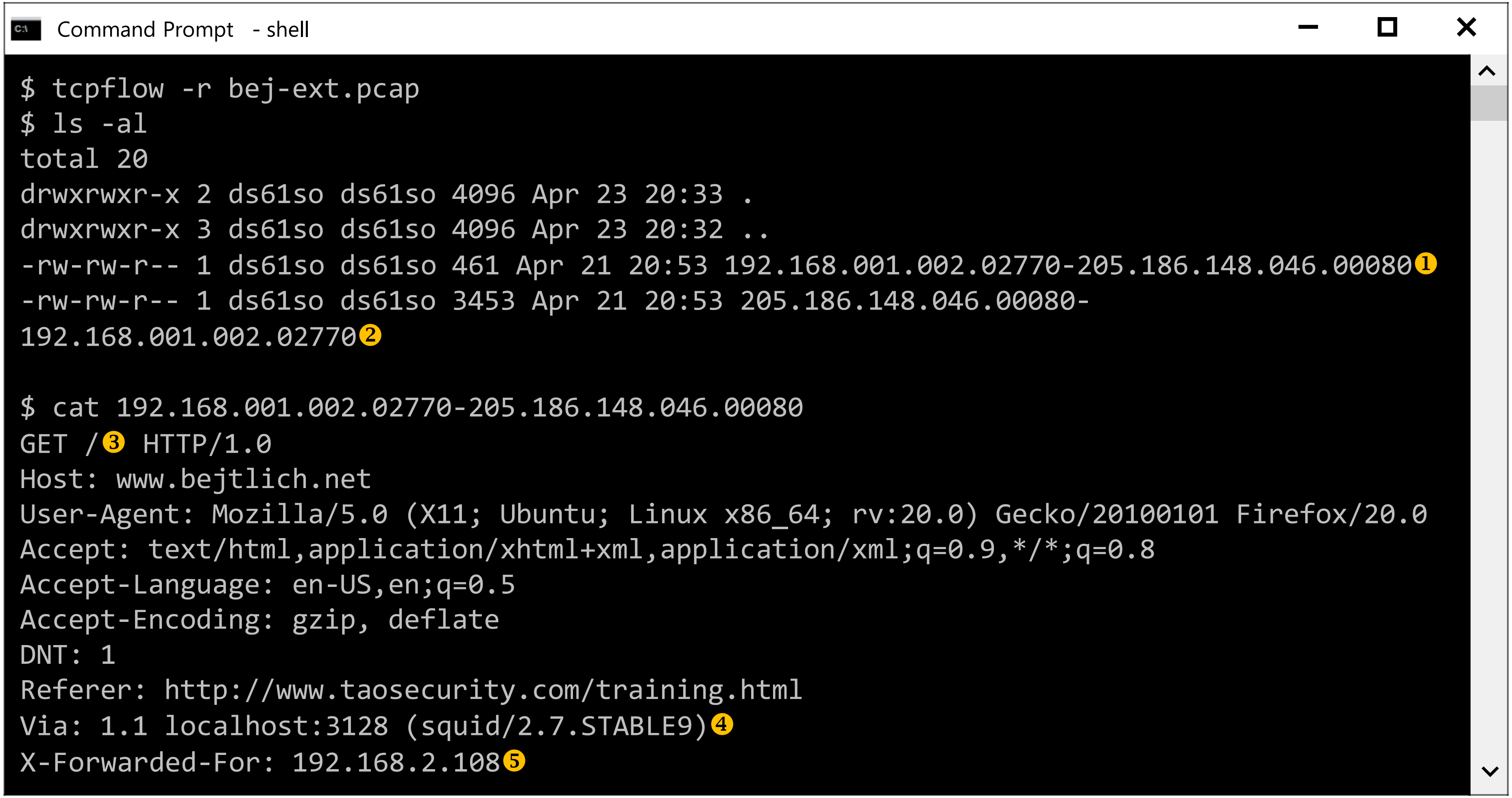
```
$ tcpflow -r bej-int.pcap
$ ls -al
total 56
drwxrwxr-x 3 ds61so ds61so 4096 Apr 23 20:14 .
drwxrwxr-x 4 ds61so ds61so 4096 Apr 23 20:05 ..
-rw-rw-r-- 1 ds61so ds61so 3605 Apr 21 20:53 172.016.002.001.03128-192.168.002.108.50949❶
-rw-rw-r-- 1 ds61so ds61so 376 Apr 21 20:53 192.168.002.108.50949-172.016.002.001.03128❷



$ cat 192.168.002.108.50949-172.016.002.001.03128
GET http://www.bejtlich.net/❸ HTTP/1.1
Host: www.bejtlich.net
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:20.0) Gecko/20100101 Firefox/20.0
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
DNT: 1
Referer: http://www.taosecurity.com/training.html
Connection: keep-alive
```

Proxy Example – Client to Proxy



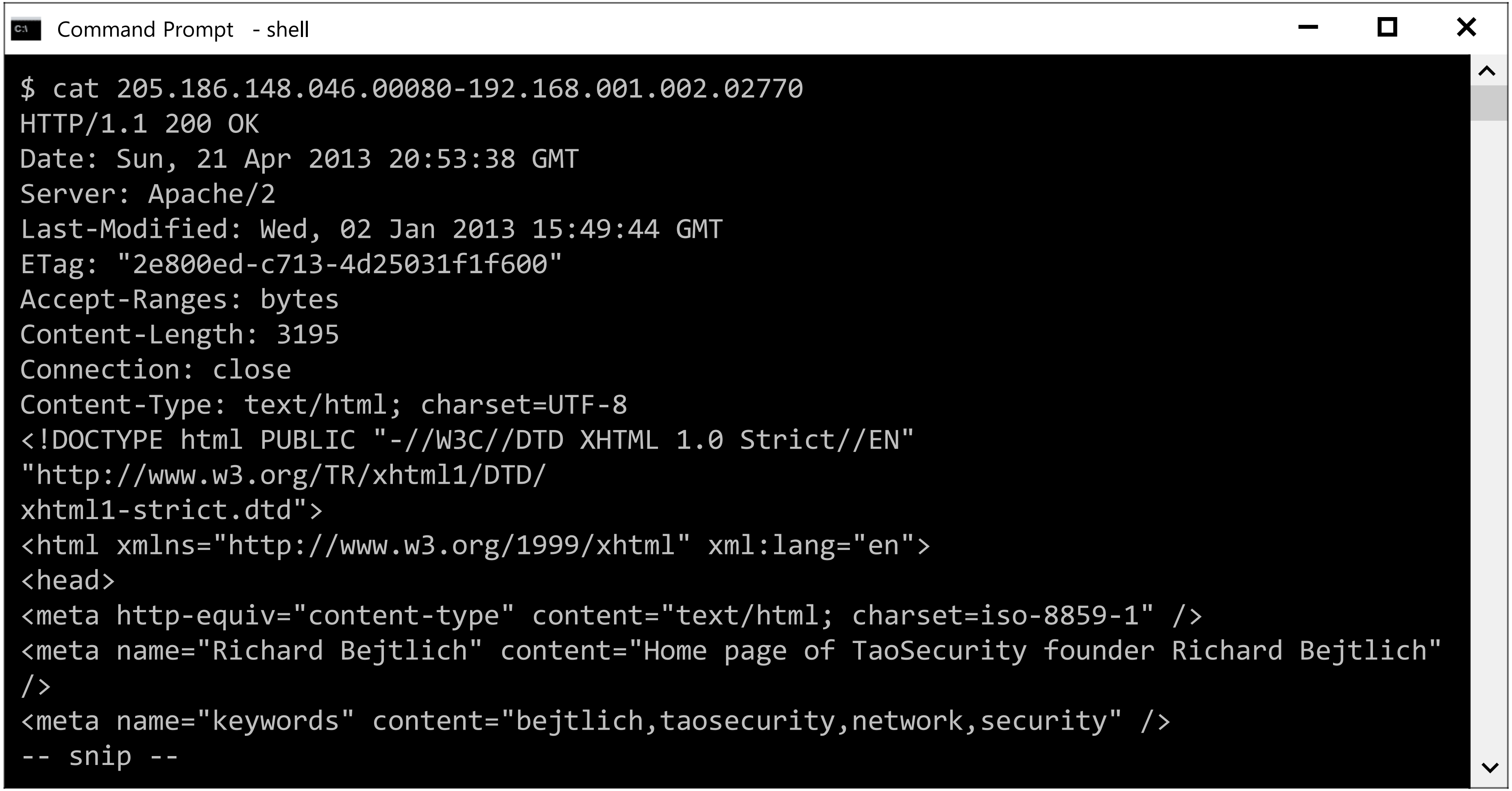
```
$ cat 172.016.002.001.03128-192.168.002.108.50949
HTTP/1.0 200 OK
Date: Sun, 21 Apr 2013 20:53:38 GMT
Server: Apache/2
Last-Modified: Wed, 02 Jan 2013 15:49:44 GMT
ETag: "2e800ed-c713-4d25031f1f600"
Accept-Ranges: bytes
Content-Length: 3195
Content-Type: text/html; charset=UTF-8
X-Cache: MISS from localhost❶
X-Cache-Lookup: MISS from localhost:3128❷
Via: 1.1 localhost:3128 (squid/2.7.STABLE9)❸
Connection: keep-alive
Proxy-Connection: keep-alive❹
❺<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/
xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
-- snip --
```

Proxy Example – Proxy to Server



```
$ tcpflow -r bej-ext.pcap
$ ls -al
total 20
drwxrwxr-x 2 ds61so ds61so 4096 Apr 23 20:33 .
drwxrwxr-x 3 ds61so ds61so 4096 Apr 23 20:32 ..
-rw-rw-r-- 1 ds61so ds61so 461 Apr 21 20:53 192.168.001.002.02770-205.186.148.046.00080❶
-rw-rw-r-- 1 ds61so ds61so 3453 Apr 21 20:53 205.186.148.046.00080-
192.168.001.002.02770❷

$ cat 192.168.001.002.02770-205.186.148.046.00080
GET /❸ HTTP/1.0
Host: www.bejtlich.net
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:20.0) Gecko/20100101 Firefox/20.0
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
DNT: 1
Referer: http://www.taosecurity.com/training.html
Via: 1.1 localhost:3128 (squid/2.7.STABLE9)❹
X-Forwarded-For: 192.168.2.108❺
```

Proxy Example – Proxy to Server

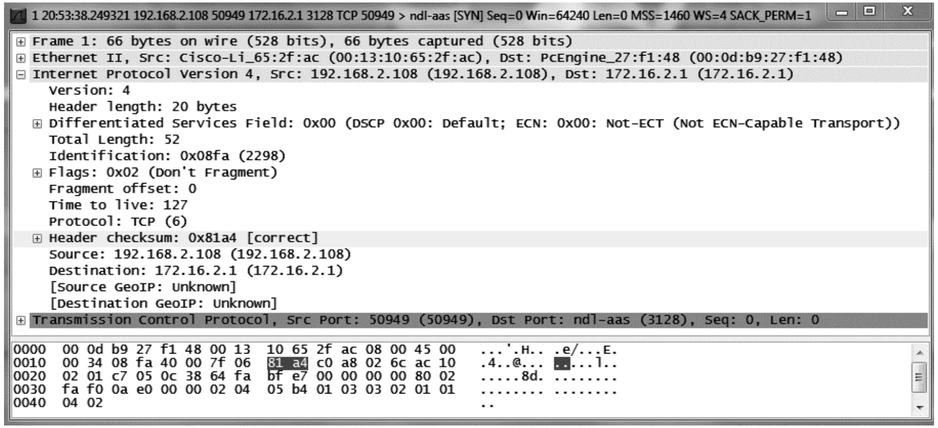


```
$ cat 205.186.148.046.00080-192.168.001.002.02770
HTTP/1.1 200 OK
Date: Sun, 21 Apr 2013 20:53:38 GMT
Server: Apache/2
Last-Modified: Wed, 02 Jan 2013 15:49:44 GMT
ETag: "2e800ed-c713-4d25031f1f600"
Accept-Ranges: bytes
Content-Length: 3195
Connection: close
Content-Type: text/html; charset=UTF-8
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/
xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
<meta name="Richard Bejtlich" content="Home page of TaoSecurity founder Richard Bejtlich"
/>
<meta name="keywords" content="bejtlich,taosecurity,network,security" />
-- snip --
```

Checksums

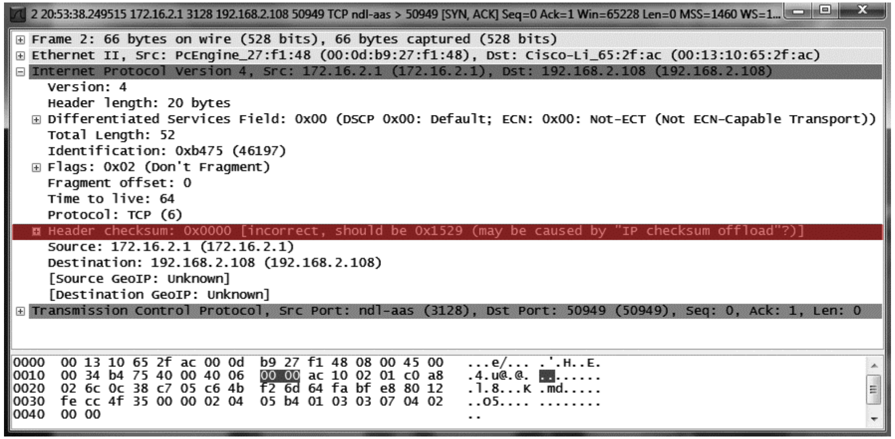
- IP headers contain a checksum to detect errors or corruption in the IP header
- Network devices recalculate this checksum when the packet is processed
- If the calculated checksum does not match that listed in the packet, the packet will be dropped

Identifying Checksums with Wireshark



1 20:53:38.249321 192.168.2.108 50949 172.16.2.1 3128 TCP 50949 > ndl-aas [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=4 SACK_PERM=1

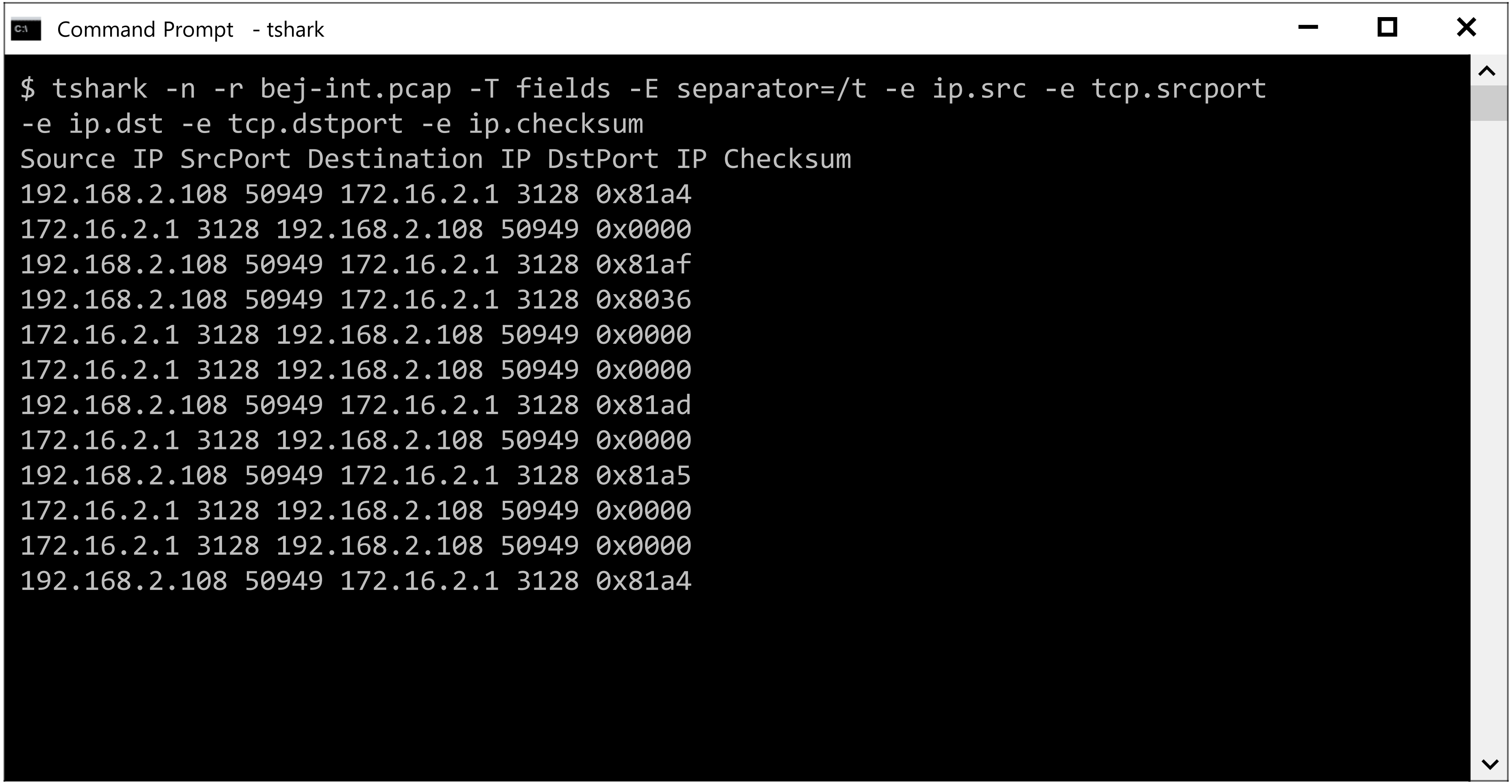
```
⊞ Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
⊞ Ethernet II, Src: Cisco-Li_65:2f:ac (00:13:10:65:2f:ac), Dst: PcEngine_27:f1:48 (00:0d:b9:27:f1:48)
⊟ Internet Protocol Version 4, Src: 192.168.2.108 (192.168.2.108), Dst: 172.16.2.1 (172.16.2.1)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 52
    Identification: 0x08fa (2298)
  ⊞ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 127
    Protocol: TCP (6)
  ⊞ Header checksum: 0x81a4 [correct]
    Source: 192.168.2.108 (192.168.2.108)
    Destination: 172.16.2.1 (172.16.2.1)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
⊞ Transmission Control Protocol, Src Port: 50949 (50949), Dst Port: ndl-aas (3128), Seq: 0, Len: 0

0000  00 0d b9 27 f1 48 00 13  10 65 2f ac 08 00 45 00   ...'.H.. .e/...E.
0010  00 34 08 fa 40 00 7f 06  81 a4 c0 a8 02 6c ac 10   .4..@... .....l..
0020  02 01 c7 05 0c 38 64 fa  bf e7 00 00 00 00 80 02   .....8d. ........
0030  fa f0 0a e0 00 00 02 04  05 b4 01 03 03 02 01 01   ........ ........
0040  04 02                                              ..
```



2 20:53:38.249515 172.16.2.1 3128 192.168.2.108 50949 TCP ndl-aas > 50949 [SYN, ACK] Seq=0 Ack=1 Win=65228 Len=0 MSS=1460 WS=1...

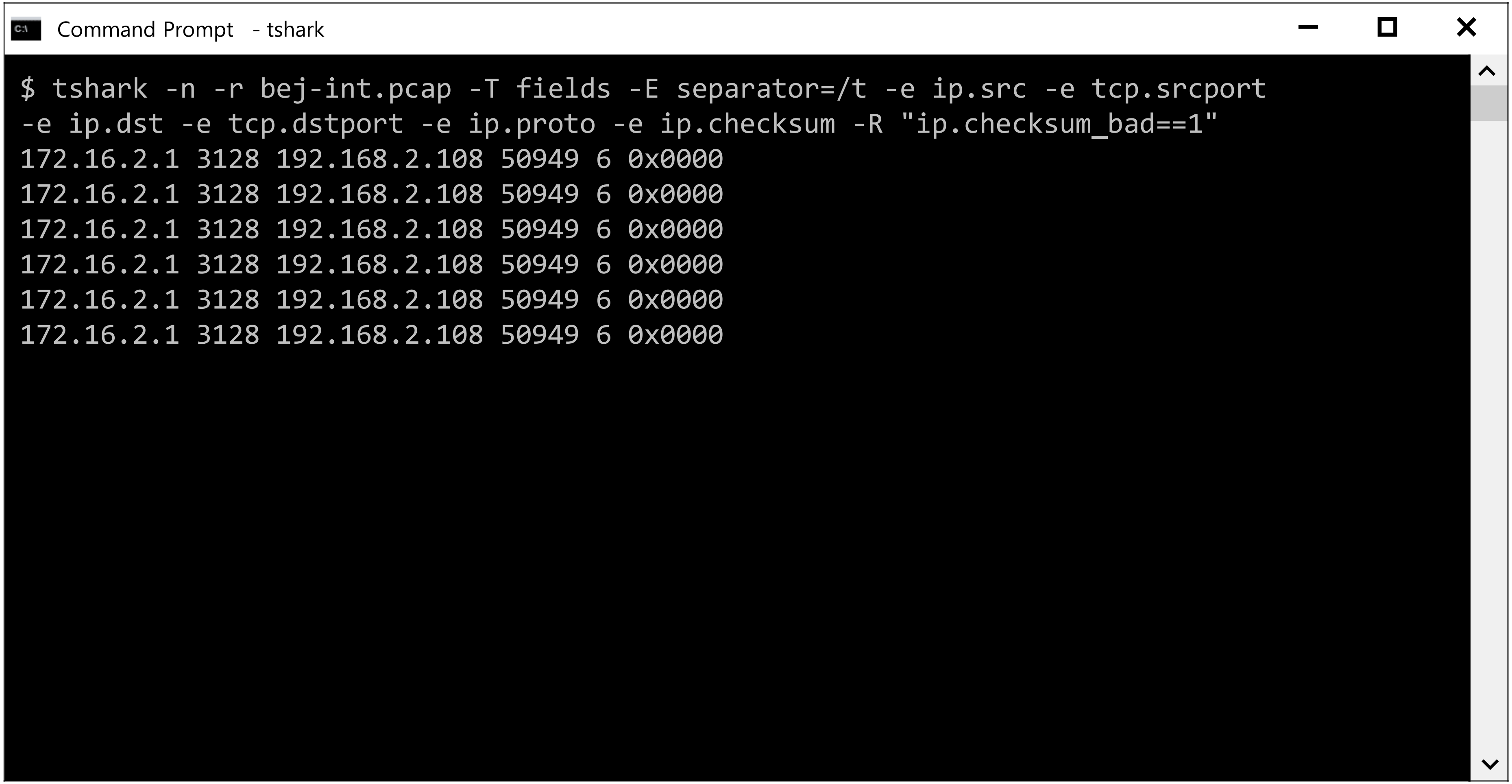
```
⊞ Frame 2: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
⊞ Ethernet II, Src: PcEngine_27:f1:48 (00:0d:b9:27:f1:48), Dst: Cisco-Li_65:2f:ac (00:13:10:65:2f:ac)
⊟ Internet Protocol Version 4, Src: 172.16.2.1 (172.16.2.1), Dst: 192.168.2.108 (192.168.2.108)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 52
    Identification: 0xb475 (46197)
  ⊞ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
  ⊞ Header checksum: 0x0000 [incorrect, should be 0x1529 (may be caused by "IP checksum offload"?)]
    Source: 172.16.2.1 (172.16.2.1)
    Destination: 192.168.2.108 (192.168.2.108)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
⊞ Transmission Control Protocol, Src Port: ndl-aas (3128), Dst Port: 50949 (50949), Seq: 0, Ack: 1, Len: 0

0000  00 13 10 65 2f ac 00 0d  b9 27 f1 48 08 00 45 00   ...e/... .'.H..E.
0010  00 34 b4 75 40 00 40 06  00 00 ac 10 02 01 c0 a8   .4.u@.@. ........
0020  02 6c 0c 38 c7 05 c6 4b  f2 6d 64 fa bf e8 80 12   .l.8...K .md.....
0030  fe cc 4f 35 00 00 02 04  05 b4 01 03 03 07 04 02   ..O5.... ........
0040  00 00                                              ..
```

Identifying Checksums with Tshark



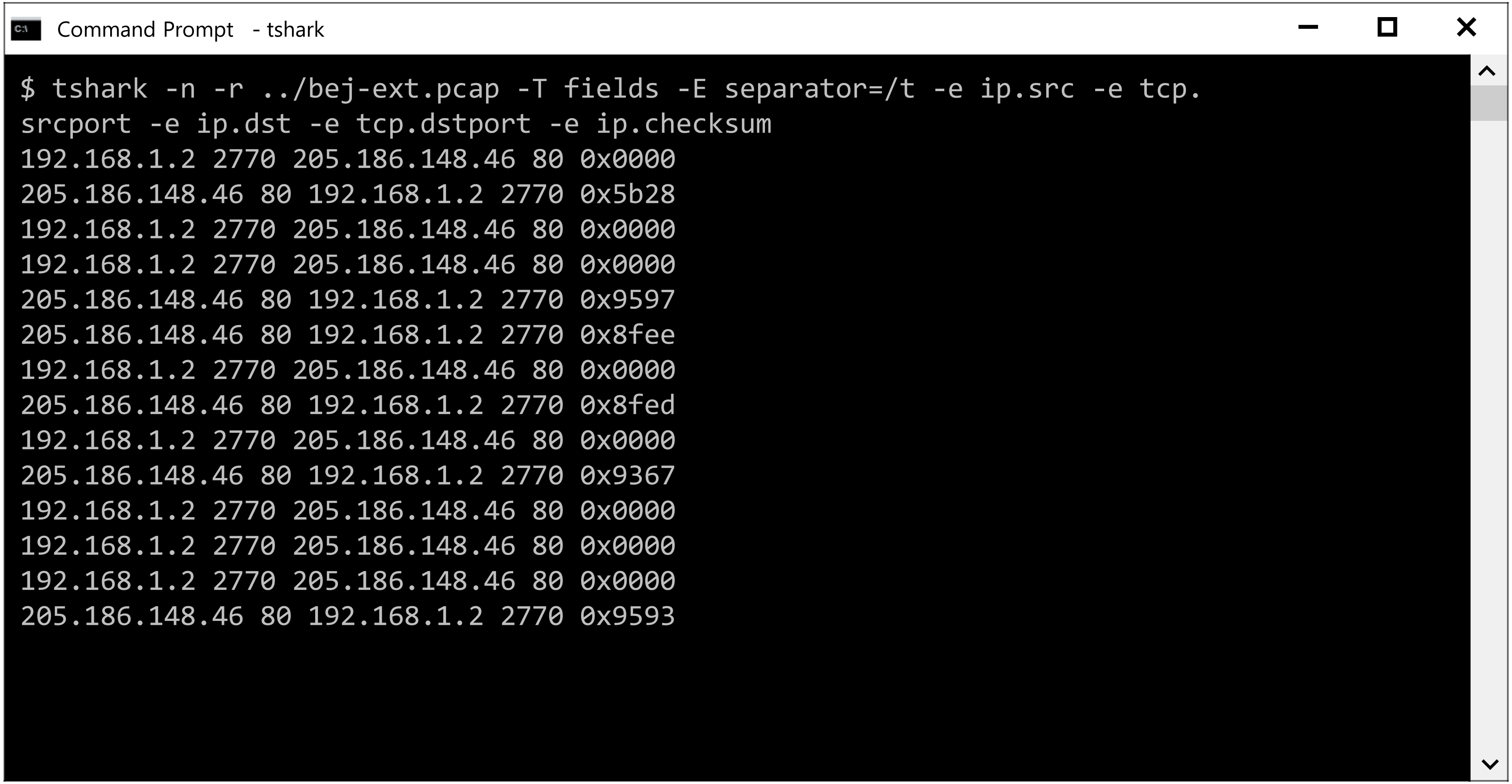
```
$ tshark -n -r bej-int.pcap -T fields -E separator=/t -e ip.src -e tcp.srcport
-e ip.dst -e tcp.dstport -e ip.checksum
Source IP SrcPort Destination IP DstPort IP Checksum
192.168.2.108 50949 172.16.2.1 3128 0x81a4
172.16.2.1 3128 192.168.2.108 50949 0x0000
192.168.2.108 50949 172.16.2.1 3128 0x81af
192.168.2.108 50949 172.16.2.1 3128 0x8036
172.16.2.1 3128 192.168.2.108 50949 0x0000
172.16.2.1 3128 192.168.2.108 50949 0x0000
192.168.2.108 50949 172.16.2.1 3128 0x81ad
172.16.2.1 3128 192.168.2.108 50949 0x0000
192.168.2.108 50949 172.16.2.1 3128 0x81a5
172.16.2.1 3128 192.168.2.108 50949 0x0000
172.16.2.1 3128 192.168.2.108 50949 0x0000
192.168.2.108 50949 172.16.2.1 3128 0x81a4
```

Identifying Checksums with Tshark



```
$ tshark -n -r bej-int.pcap -T fields -E separator=/t -e ip.src -e tcp.srcport
-e ip.dst -e tcp.dstport -e ip.proto -e ip.checksum -R "ip.checksum_bad==1"
172.16.2.1 3128 192.168.2.108 50949 6 0x0000
172.16.2.1 3128 192.168.2.108 50949 6 0x0000
172.16.2.1 3128 192.168.2.108 50949 6 0x0000
172.16.2.1 3128 192.168.2.108 50949 6 0x0000
172.16.2.1 3128 192.168.2.108 50949 6 0x0000
172.16.2.1 3128 192.168.2.108 50949 6 0x0000
```

Identifying Checksums with Tshark



```
$ tshark -n -r ../bej-ext.pcap -T fields -E separator=/t -e ip.src -e tcp.
srcport -e ip.dst -e tcp.dstport -e ip.checksum
192.168.1.2 2770 205.186.148.46 80 0x0000
205.186.148.46 80 192.168.1.2 2770 0x5b28
192.168.1.2 2770 205.186.148.46 80 0x0000
192.168.1.2 2770 205.186.148.46 80 0x0000
205.186.148.46 80 192.168.1.2 2770 0x9597
205.186.148.46 80 192.168.1.2 2770 0x8fee
192.168.1.2 2770 205.186.148.46 80 0x0000
205.186.148.46 80 192.168.1.2 2770 0x8fed
192.168.1.2 2770 205.186.148.46 80 0x0000
205.186.148.46 80 192.168.1.2 2770 0x9367
192.168.1.2 2770 205.186.148.46 80 0x0000
192.168.1.2 2770 205.186.148.46 80 0x0000
192.168.1.2 2770 205.186.148.46 80 0x0000
205.186.148.46 80 192.168.1.2 2770 0x9593
```

How Bad Checksums Happen

- Checksums occasionally fail due to errors or corruption that occur from hosts on the internet
- The errors reported by Wireshark mention IP checksum offload as a potential source
- IP checksum offload is a function of either the NIC driver, or hardware of the card in higher end models
- IP checksum offload reduces the burden of the primary CPU
- By default, SO disables driver and hardware offloading in order to avoid these issues

Zeek and Bad Checksums

- Some security tools assume that packets with a bad IP checksum will not be processed by an endpoint, and the security tool may drop the packet
- Zeek ignores traffic with bad checksums by default
- Running bro with the –C switch tells it to ignore bad checksums and process the traffic
- The best solution is to disable checksum offloading on the offending device

Summary

- Making slight modifications to the SO system and workflows can add additional visibility and improve performance
- One feature of Zeek allows the identification of malware, by comparing MD5 hash to that of known malicious files
- Submitting binary files extracted with Zeek to analysis engines provides greater context about user applications to analysts
- Zeek includes a module to identify know IoC related to APT1
- The Malware Hash Registry offers analysts a quick and easy way to identify malware by MD5 sum from the command line

Summary

- Proxies are used in corporate networks to provide filtering and security services for web traffic
- IP checksums if not configured and managed correctly can add complexity to NSM operations
- IP checksum offloading should be disabled on systems that are used to capture traffic for the purpose of NSM

References

- Bejtlich, R. (2013). Chapter 12: Extending SO. In The practice of network security monitoring understanding incident detection and response. San Francisco: No Starch Press.
- Team Cymru Inc. (n.d.). We are Team Cymru. Retrieved April 13, 2020, from https://www.team-cymru.com/aboutus.html