# Solving k3anu_evidence pcap with Zeek - Sean Yarkoni

Solving the challenge k3anu_evidence with Zeek on Ubuntu-20.04.2-live-server-amd64.

**The challenges are:**

**1)** What is the hostname of the system the PCAP was recovered from? (all caps)

**2)** What is the user agent K3anu use?

**3)** What operating system did K3anu use?

**4)** How many different DNS queries are in the PCAP file?

**5)** How many DNS queries in the PCAP received NXdomain?

**6)** What IP and port does the executable came from?

**7)** What is the name of the executable that was sent?

**8)** What can be find using different Zeek signature?

PCAP file working with can be found in:
https://drive.google.com/file/d/1bimpl4aHw25n87On47fMqQQsTX168gna/view

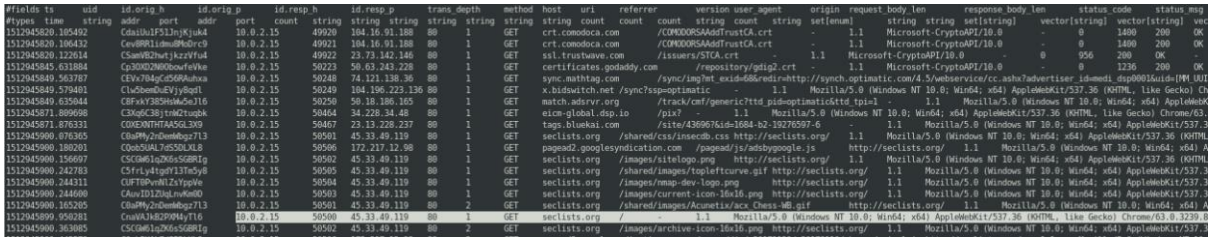
**Solution:**

In order for us to start we need Zeek to read the PCAP file. For that will need to type: zeek -r k3anu_evidence.pcapng -> This command will tell Zeek to read the file and create different Log files that Zeek can create using the information that can be found in the PCAP file.


Now we can start solving the challenge.

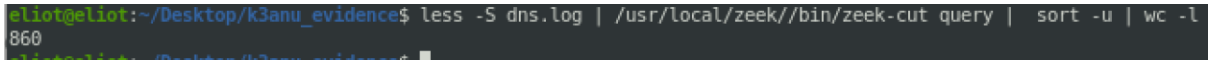
**1)** The hostname of the system the PCAP was recovered from can be found in the dhcp.log file because when a computer need to get an IP address it needs to identify itself in front of the DHCP server, we can see the hostname by typing: less -S dhcp.log
and the hostname is: MSEDGEWIN10.

**2)** The user agent K3anu using can be found in the http.log file under the user agent column and we can find it by using the IP address found in challenge 1 from the dhcp.log file IP-10.0.2.15 by typing: less -S http.log we can see that the user agent is: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.84 Safari/537.36 .

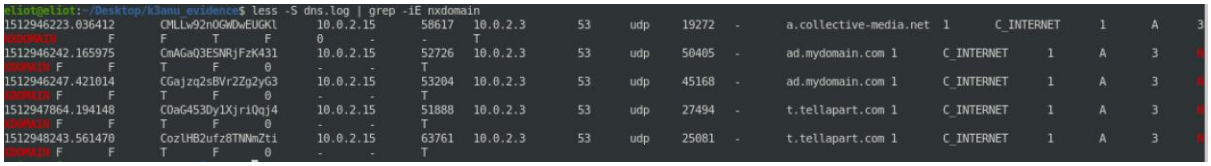


**3)** The operating system K3anu use can be seen in the user agent we found in challenge 2 (Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.84 Safari/537.36) and the operating system is: Windows NT 10.0; Win64; x64.

**4)** The amount of the DNS queries that are in the PCAP file can be found by using Zeek-cut on the dns.log file and by organizing the results, we need to type:
less -S dns.log | /usr/local/zeek//bin/zeek-cut query | sort -u | wc -l and then we can see that 860 queries were made in the PCAP file.

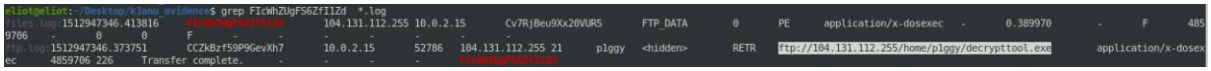



**5)** We can find how many DNS queries NXdomain received in the PCAP file by locking at the file dns.log and grepping NXdomain, we will type: less -S dns.log | grep -iE nxdomain and see that there are 5 DNS queries in the PCAP file to NXdomain




**6)** The IP and port the executable came from can be found by looking at the pe.log file that contain portable executable information by typing: less -S pe.log we can see that Zeek found only 1 executable in the PCAP file, we can grep the id string Zeek gave to that portable executable and find more information on the file by typing: grep FIcWhZUgFS6ZfI1Zd *.log , now we can see that the file came from 104.131.112.255 on port 21


**7)** The name of the executable that was sent can be found using what we found in the previous challenge (challenge 6) and if we type: grep FIcWhZUgFS6ZfI1Zd *.log we can see that there is a file sent from 104.131.112.255/home/p1ggy and its called decrypttool.exe which is the executable we were trying to find and as we know from the pe.log it is the only executable in the PCAP file.




**8)** Zeek signature are very versatile option for us to find different information that Zeek can't find when we run it on the PCAP file with the "out of the box" options, in order for us to use signature we will need to create a directory and a file that is ".sig" in /zeek/scripts/base/protocols/ with the command:

sudo mkdir <name>

cd <name>

sudo touch <name>.sig

now we need to edit the file as sudo with any text editor (I'm using "vim") and add the signature we want.

**A)** In the challenges we just solve we saw different files and a lot of transactions between different IP addresses if we want to focus on 1 IP address, we can use Zeek signature to isolate the IP address we are after, in order to do so we will type:

Sudo vim <name>.sig

And in the file:

signature <name> {

ip-proto == tcp

header ip[16:4] == 104.131.112.255/30

payload /./

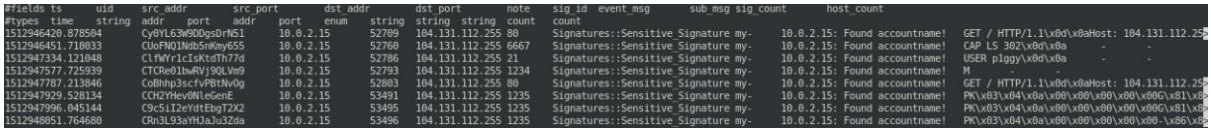
event "<name>"

}

Now we can go back to the PCAP file and read it with Zeek and add the signature file we want and type:

zeek -r k3anu_evidence.pcapng -s /home/eliot/zeek/scripts/base/protocols/test/test.sig

Now we can go to the new file created signatures.log and see that the file contains connections only from 104.131.112.255 that containing data (if we want only the connections without any information we can go to conn.log or delete the line: payload /./ )




* The data we see now is a representation of python bytes a method that convert data into immutable sequence (cannot be modified), more on the subject at:
https://www.programiz.com/python-programming/methods/built-in/bytes

**B)** We can use Zeek signature to view a port that might interest us and the related data on that port we can use the signatures to view more information or data that Zeek don't show us with the "out of the box" settings. For example, on port 80 we can see http request look like this:

If we want to get more organized information we can go to the ".sig" file we created at the start of this challenge and type:

Sudo vim <name>.sig

And in the file:

signature <name> {

ip-proto == tcp

dst-port == 80

http-request //

http-request-header //

event "<name>"

}

Now we can go back to the PCAP file and read it with Zeek and add the signature file and type:

zeek -r k3anu_evidence.pcapng -s /home/eliot/zeek/scripts/base/protocols/test/test.sig

Now we can go to the new file created signatures.log and see that the file contains only connections on port 80 but with the same data as the http.log file.

If we want to get more information (if possible) we can go back to the .sig file and edit it to give us different information, type:

Sudo vim /nano <name>.sig

```
signature <name> {
    ip-proto == tcp
    dst-port == 80
   http-request-body  // <*Note: instead, you can use: http-reply-body >
    event "<name>"
}
```

Now if we read the PCAP file again with the command:

zeek -r k3anu_evidence.pcapng -s /home/eliot/zeek/scripts/base/protocols/test/test.sig

We can see in the new signatures.log file that the result contains information only on port 80 but information we could not see before:


**How Zeek help us in the k3anu_evidence case?**

**1)** We toke the k3anu_evidence.pcapng file and run it throw Zeek to get a sorted Log files of the traffic captured in the pcap file by category.

**2)** We looked at all the files and figure what they contain before getting in to work.

Conn.log is a file containing TCP/UDP/ICMP connections.

Dhcp.log is a file containing dynamic host configuration protocol leases.

Dns.log is a file containing domain name system activity.

Files.log is a file containing file analysis results.

Ftp.log is a file containing file transfer protocol activity.

Http.log is a file containing hypertext transfer protocol requests and replies.

Irc.log is a file containing internet relay chat commands and responses.

Packet_filter.log is a file containing list of packet filters that were applied.

Pe.log is a file containing portable executable

Ssl.log is a file containing SSL/TLS handshake info.

Weird.log is a file containing session initiation protocol.

x509.log is a file containing X.509 certificate info.

**3)** We saw that the log files Zeek generate show us rich related information that can be found in each packet wan Zeek read the pcap file with his "out of the box" settings, each file is sorted in a way that it is easy to read and understand in order for us to find what we need easily.

**4)** We saw that Zeek give's packets a string that we can copy and grep to find related information connected to the packet from different log files and it will be easier to us to "dig" through a suspicions or interesting connection.

**5)** We experience with Zeek a very versatile signatures that we can make in order to get the information we are after, but in order to get that signature right and the information relevant, we need to know in which protocol and layer the information we want to get is supposed to be, when using signature, the user knowledge have a big role.