Lucas McMahon
12/14/23

Zeek Scripting Notes

Goal

To gain proficiency in intrusion detection using Zeek, learning its event-driven scripting language for investigating and correlating detected events. The training covers script basics, signature application, and frameworks, offering a comprehensive understanding of Zeek's capabilities in intrusion detection.

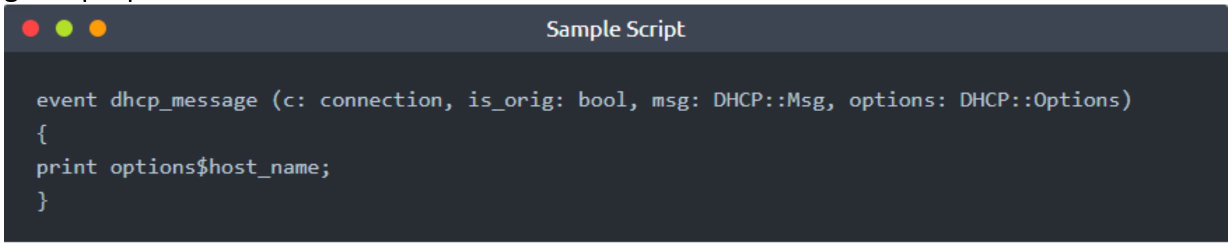
Zeek has its own event-driven scripting language, which is as powerful as high-level languages and allows us to investigate and correlate the detected events. Since it is as capable as high-level programming languages, you will need to spend time on Zeek scripting language in order to become proficient. In this room, we will cover the basics of Zeek scripting to help you understand, modify and create basic scripts. Note that scripts can be used to apply a policy and in this case, they are called policy scripts.

From <https://tryhackme.com/room/zeekbro>

Task 1: Scripts

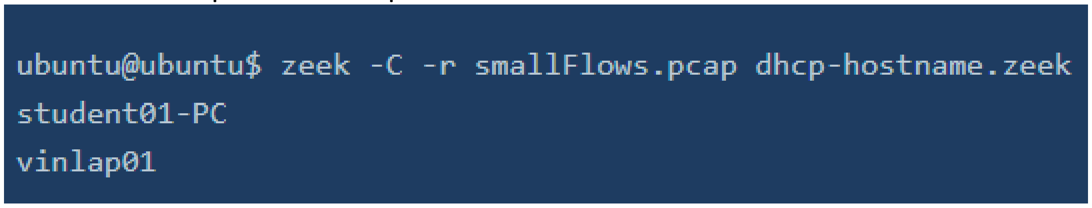
One powerful advantage of Zeek compared to tools like Wireshark is the ability to automate tasks using scripts. Transferring data between applications like Wireshark and other applications for processing require manual work.

In this sample script, the third line tells Zeek to extract any DHCP hostnames found in a given pcap file.



```
event dhcp_message (c: connection, is_orig: bool, msg: DHCP::Msg, options: DHCP::Options)
{
print options$host_name;
}
```

Here is an example of the script in use.



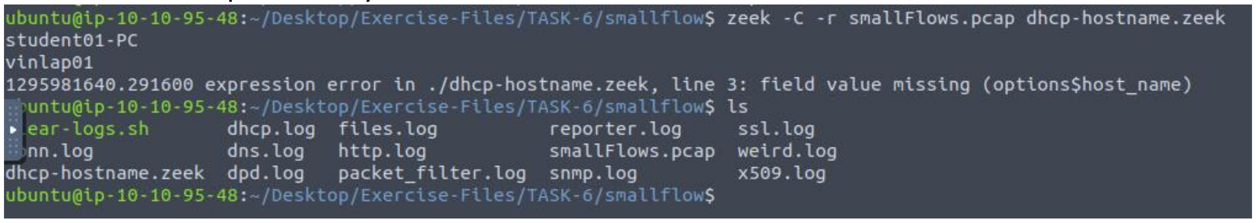
```
ubuntu@ubuntu$ zeek -C -r smallFlows.pcap dhcp-hostname.zeek
student01-PC
vinlap01
```

This tool can be used to locate any rogue users on a device automatically to potentially prevent an attack. My next questions will put it into practical use.

Question 1:

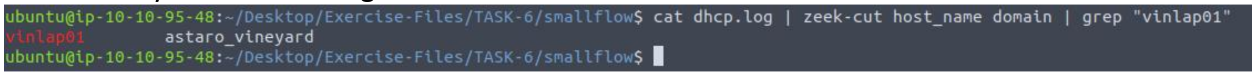
Investigate the smallFlows.pcap file. Investigate the dhcp.log file. What is the domain value of the "vinlap01" host?

I ran this script against a given pcap file with a goal in mind of finding the domain value of the host "vinlap01" on my network.



```
ubuntu@ip-10-10-95-48:~/Desktop/Exercise-Files/TASK-6/smallflow$ zeek -C -r smallFlows.pcap dhcp-hostname.zeek
student01-PC
vinlap01
1295981640.291600 expression error in ./dhcp-hostname.zeek, line 3: field value missing (options$host_name)
untu@ip-10-10-95-48:~/Desktop/Exercise-Files/TASK-6/smallflow$ ls
ear-logs.sh       dhcp.log  files.log          reporter.log     ssl.log
nn.log            dns.log   http.log           smallFlows.pcap  weird.log
dhcp-hostname.zeek  dpd.log   packet_filter.log  snmp.log         x509.log
ubuntu@ip-10-10-95-48:~/Desktop/Exercise-Files/TASK-6/smallflow$
```

Using what I learned before, I can investigate the dhcp.log file and cut out the unnecessary information it gives me.



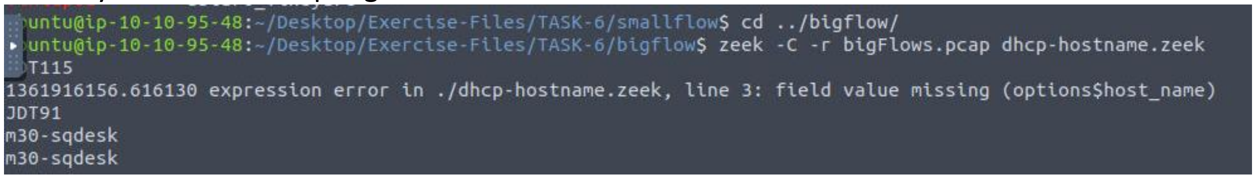
```
ubuntu@ip-10-10-95-48:~/Desktop/Exercise-Files/TASK-6/smallflow$ cat dhcp.log | zeek-cut host_name domain | grep "vinlap01"
vinlap01        astaro_vineyard
ubuntu@ip-10-10-95-48:~/Desktop/Exercise-Files/TASK-6/smallflow$
```

The domain name I am looking for was *astaro_vineyard*

Question 2:

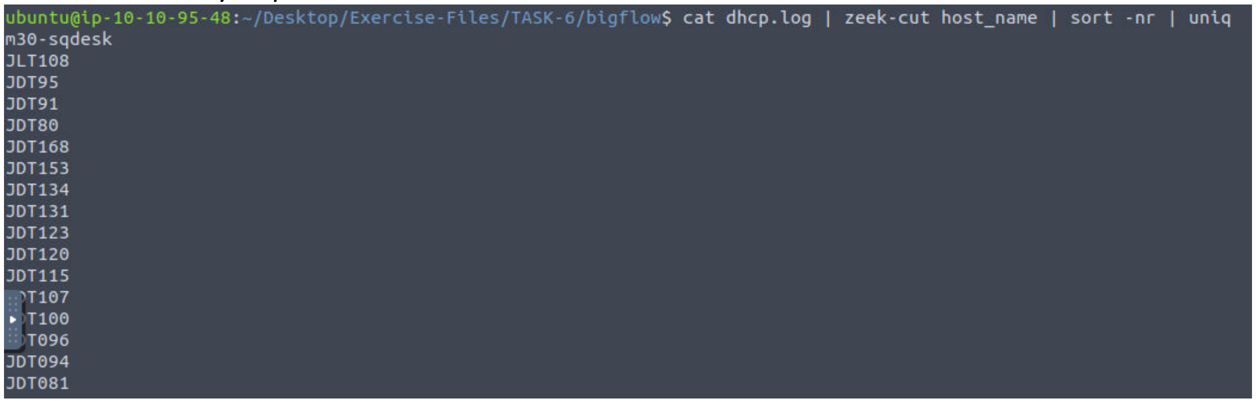
Investigate the bigFlows.pcap file. Investigate the dhcp.log file. What is the number of identified unique hostnames?

I ran my DHCP-Zeek script against this file like last time.



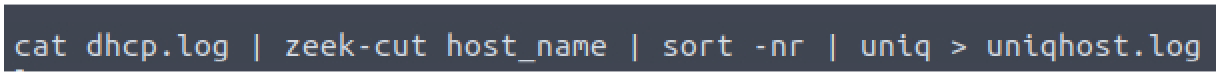
```
untu@ip-10-10-95-48:~/Desktop/Exercise-Files/TASK-6/smallflow$ cd ../bigflow/
untu@ip-10-10-95-48:~/Desktop/Exercise-Files/TASK-6/bigflow$ zeek -C -r bigFlows.pcap dhcp-hostname.zeek
T115
1361916156.616130 expression error in ./dhcp-hostname.zeek, line 3: field value missing (options$host_name)
JDT91
m30-sqdesk
m30-sqdesk
```

I ran a cat command which also removed any information other than the host name, and removed any duplicates.

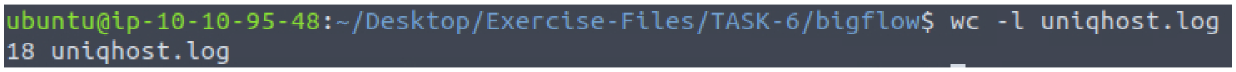


```
ubuntu@ip-10-10-95-48:~/Desktop/Exercise-Files/TASK-6/bigflow$ cat dhcp.log | zeek-cut host_name | sort -nr | uniq
m30-sqdesk
JLT108
JDT95
JDT91
JDT80
JDT168
JDT153
JDT134
JDT131
JDT123
JDT120
JDT115
T107
T100
T096
JDT094
JDT081
```

I am able to count the number of hostnames, but I figured if given another file much larger than this one, I should be able to know how to get this count without doing it manually. So I made a file based on the output and had it tell me how many lines there were.



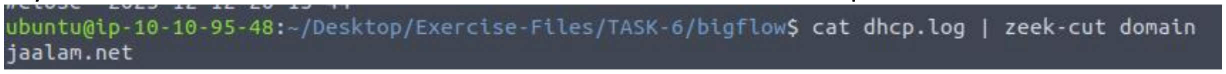
```
cat dhcp.log | zeek-cut host_name | sort -nr | uniq > uniqhost.log
```



```
ubuntu@ip-10-10-95-48:~/Desktop/Exercise-Files/TASK-6/bigflow$ wc -l uniqhost.log
18 uniqhost.log
```

Question 3:
Investigate the ***dhcp.log*** file. What is the identified domain value?

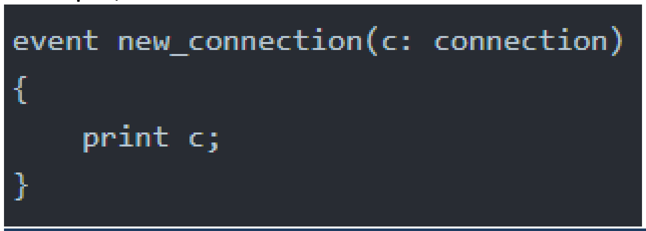
My last task for this room was to find the other domain name responsible for this traffic



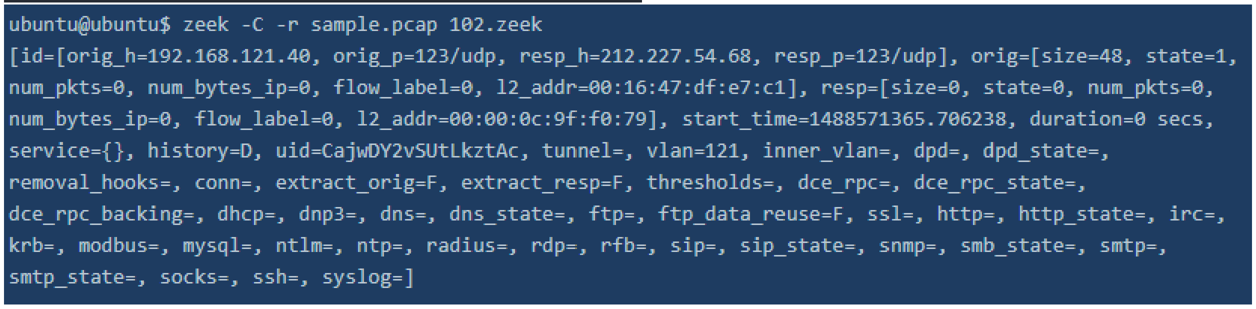
```
ubuntu@ip-10-10-95-48:~/Desktop/Exercise-Files/TASK-6/bigflow$ cat dhcp.log | zeek-cut domain
jaalam.net
```

<u>Task 2: Signatures</u>

The start of my next task outlines the basics for creating a script in Zeek. In a given example, I can see how to find the contents of a new connection in a pcap file.

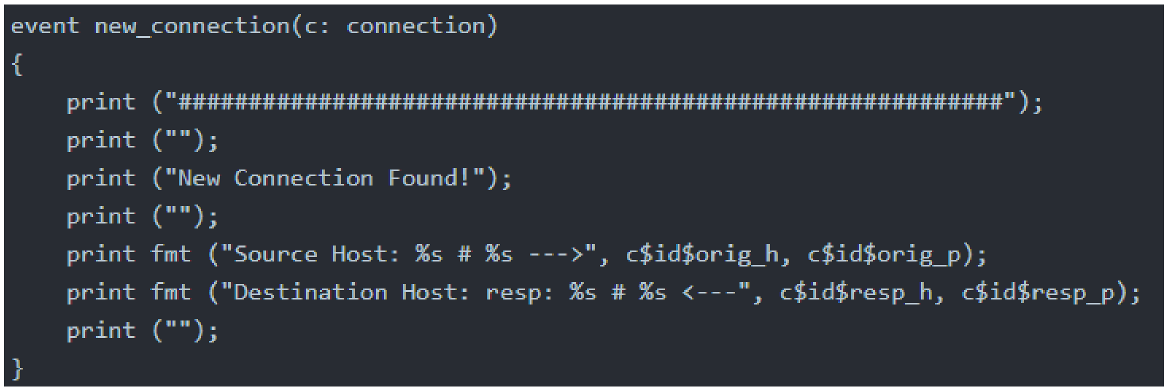


```
event new_connection(c: connection)
{
    print c;
}
```

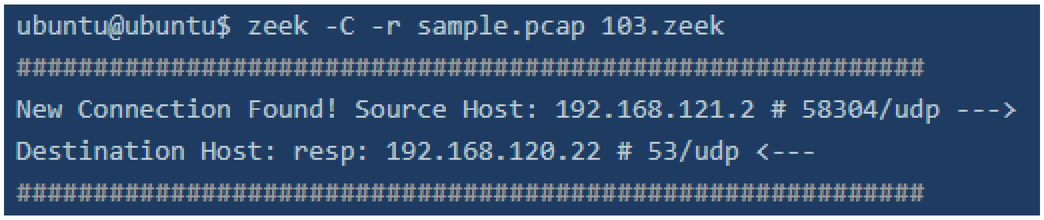


```
ubuntu@ubuntu$ zeek -C -r sample.pcap 102.zeek
[id=[orig_h=192.168.121.40, orig_p=123/udp, resp_h=212.227.54.68, resp_p=123/udp], orig=[size=48, state=1,
num_pkts=0, num_bytes_ip=0, flow_label=0, l2_addr=00:16:47:df:e7:c1], resp=[size=0, state=0, num_pkts=0,
num_bytes_ip=0, flow_label=0, l2_addr=00:00:0c:9f:f0:79], start_time=1488571365.706238, duration=0 secs,
service={}, history=D, uid=CajwDY2vSUtLkztAc, tunnel=, vlan=121, inner_vlan=, dpd=, dpd_state=,
removal_hooks=, conn=, extract_orig=F, extract_resp=F, thresholds=, dce_rpc=, dce_rpc_state=,
dce_rpc_backing=, dhcp=, dnp3=, dns=, dns_state=, ftp=, ftp_data_reuse=F, ssl=, http=, http_state=, irc=,
krb=, modbus=, mysql=, ntlm=, ntp=, radius=, rdp=, rfb=, sip=, sip_state=, snmp=, smb_state=, smtp=,
smtp_state=, socks=, ssh=, syslog=]
```

The contents of the output file are pretty hard to understand. But next I can see what a better written script would look like, which gives me the source IP, destination IP and the port used.

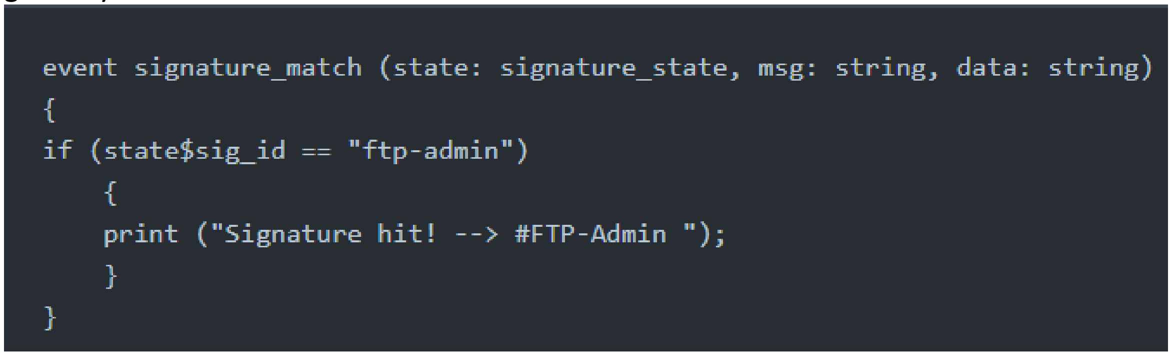


```
event new_connection(c: connection)
{
    print ("###########################################################");
    print ("");
    print ("New Connection Found!");
    print ("");
    print fmt ("Source Host: %s # %s --->", c$id$orig_h, c$id$orig_p);
    print fmt ("Destination Host: resp: %s # %s <---", c$id$resp_h, c$id$resp_p);
    print ("");
}
```



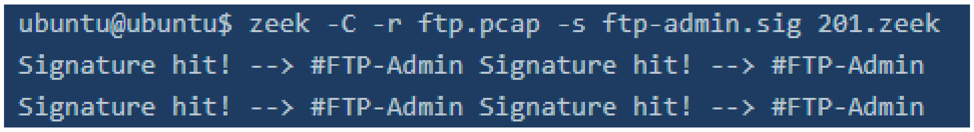
```
ubuntu@ubuntu$ zeek -C -r sample.pcap 103.zeek
###########################################################
New Connection Found! Source Host: 192.168.121.2 # 58304/udp --->
Destination Host: resp: 192.168.120.22 # 53/udp <---
###########################################################
```

Much cleaner. Now to put this into practical use, I can use some pre-defined signatures given by Zeek to let me know when certain events occur.



```
  event signature_match (state: signature_state, msg: string, data: string)
  {
  if (state$sig_id == "ftp-admin")
      {
      print ("Signature hit! --> #FTP-Admin ");
      }
  }
```

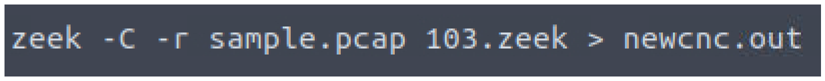
In this script, it takes in a parameter *signature_state*, which is a name for a pre written signature. In this case, I am checking if the ID of this signature is *ftp-admin* which detects FTP activity with the admin user. Running this with an example file, I can see what it looks like in action.



```
ubuntu@ubuntu$ zeek -C -r ftp.pcap -s ftp-admin.sig 201.zeek
Signature hit! --> #FTP-Admin Signature hit! --> #FTP-Admin
Signature hit! --> #FTP-Admin Signature hit! --> #FTP-Admin
```

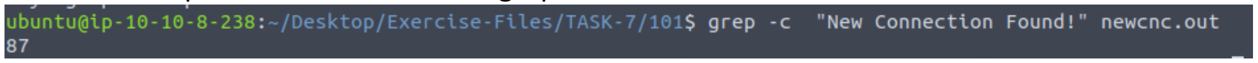
Question 1:
Investigate the sample.pcap file with 103.zeek script. Investigate the terminal output. What is the number of the detected new connections?

Now to put what I learned into practice and answer problems, first I am going to find out how many new connections occur in a given pcap file. I am also given the script, and I run it as seen.



```
zeek -C -r sample.pcap 103.zeek > newcnc.out
```

I send the output to a file so I can use grep to count the number of new connections.

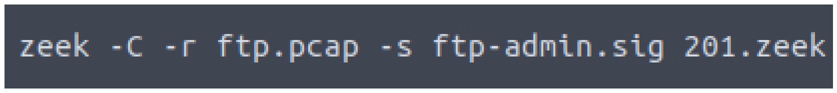


```
ubuntu@ip-10-10-8-238:~/Desktop/Exercise-Files/TASK-7/101$ grep -c  "New Connection Found!" newcnc.out
87
```

The answer I am looking for is 87.

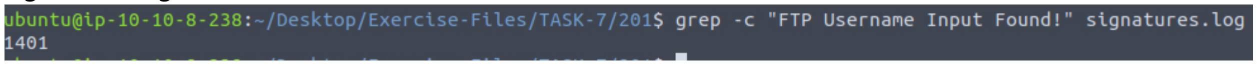
Question 2:

Investigate the pcap file with ftp-admin.sig signature and 201.zeek script. Investigate the signatures.log file. What is the number of signature hits?



```
zeek -C -r ftp.pcap -s ftp-admin.sig 201.zeek
```

Using the pre-defined *ftp-admin* signature, this command produces a log file *signature.log*.

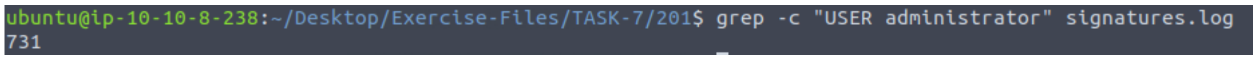


```
ubuntu@ip-10-10-8-238:~/Desktop/Exercise-Files/TASK-7/201$ grep -c "FTP Username Input Found!" signatures.log
1401
```

Using grep, the answer I am looking for is 1401.

Question 3:

Investigate the signatures.log file. What is the total number of "administrator" username detections?



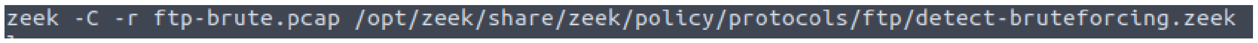
```
ubuntu@ip-10-10-8-238:~/Desktop/Exercise-Files/TASK-7/201$ grep -c "USER administrator" signatures.log
731
```

Using grep again, I find 731 matches.

Question 4:

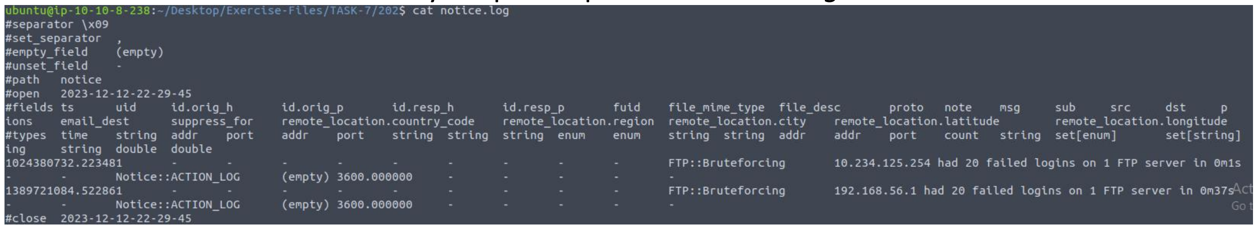
Investigate the ftp-brute.pcap file with "/opt/zeek/share/zeek/policy/protocols/ftp/detect-bruteforcing.zeek" script. Investigate the notice.log file. What is the total number of brute-force detections?

My last question asks me to find the number of brute force attempts with one of zeek's pre-defined scripts



```
zeek -C -r ftp-brute.pcap /opt/zeek/share/zeek/policy/protocols/ftp/detect-bruteforcing.zeek
```

The command runs the necessary script and produces *notice.log*.



```
ubuntu@ip-10-10-8-238:~/Desktop/Exercise-Files/TASK-7/202$ cat notice.log
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   notice
#open   2023-12-12-22-29-45
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       fuid    file_mime_type  file_desc       proto   note    msg     sub     src     dst     p
ions    email_dest      suppress_for    remote_location.country_code    remote_location.region  remote_location.city    remote_location.latitude        remote_location.longitude
#types  time    string  addr    port    addr    port    string  string  string  enum    enum    string  string  addr    addr    port    count   string  set[enum]       set[string]
ing     string  double  double
1024380732.223481       -       -       -       -       -       -       -       -       -       FTP::Bruteforcing       10.234.125.254 had 20 failed logins on 1 FTP server in 0m1s
-       -       Notice::ACTION_LOG      (empty) 3600.000000     -       -       -       -       -       -
1389721084.522861       -       -       -       -       -       -       -       -       -       FTP::Bruteforcing       192.168.56.1 had 20 failed logins on 1 FTP server in 0m37s
-       -       Notice::ACTION_LOG      (empty) 3600.000000     -       -       -       -       -       -
#close  2023-12-12-22-29-45
```

There were 2 logs created from the given file.

<u>Task 3: Frameworks</u>

Zeek has 15+ frameworks that help analysts to discover the different events of interest. In this task, we will cover the common frameworks and functions.

From <https://tryhackme.com/room/zeekbro>

One example of a Zeek framework is the *File Analysis framework*, which can call on scripts like the *hash-all-files* script. The script can locate any files and creates hash values that are asked for. Here is an example of the framework in use.

```
ubuntu@ubuntu$ zeek -C -r case1.pcap /opt/zeek/share/zeek/policy/frameworks/files/hash-all-files.zeek

ubuntu@ubuntu$ cat files.log | zeek-cut md5 sha1 sha256
cd5a4d3fdd5bffc16bf959ef75cf37bc    33bf88d5b82df3723d5863c7d23445e345828904
6137f8db2192e638e13610f75e73b9247c05f4706f0afd1fdb132d86de6b4012
b5243ec1df7d1d5304189e7db2744128    a66bd2557016377dfb95a87c21180e52b23d2e4e
f808229aa516ba134889f81cd699b8d246d46d796b55e13bee87435889a054fb
cc28e40b46237ab6d5282199ef78c464    0d5c820002cf93384016bd4a2628dcc5101211f4
749e161661290e8a2d190b1a66469744127bc25bf46e5d0c6f2e835f4b92db18
```

Another example of a framework in use is the *extract-all-files* framework that picks out any files found in a pcap file. Here is it in use.

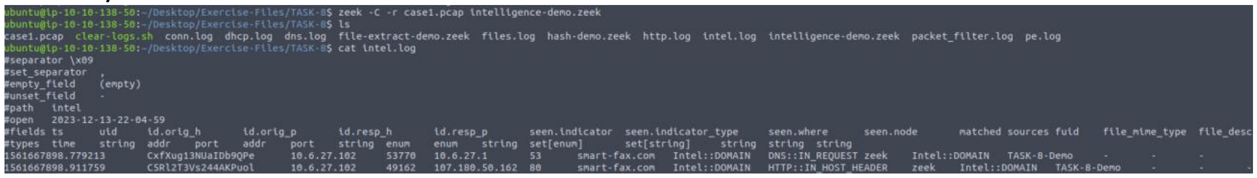
```
ubuntu@ubuntu$ zeek -C -r case1.pcap /opt/zeek/share/zeek/policy/frameworks/files/extract-all-files.zeek

ubuntu@ubuntu$ ls
101.zeek  102.zeek  103.zeek  case1.pcap  clear-logs.sh  conn.log  dhcp.log  dns.log  extract_files
files.log  ftp.pcap  http.log  packet_filter.log  pe.log
```

The command creates a separate folder called *extract_files* with all of the content.
The intelligence framework works with data to process and correlate events to identify anomalies.

Question 1:

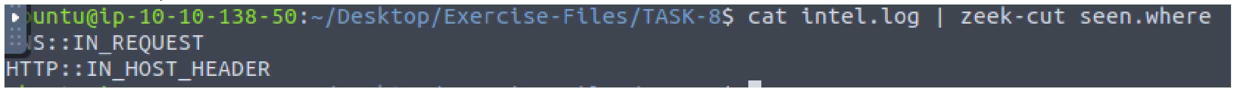
Investigate the case1.pcap file with intelligence-demo.zeek script. Investigate the intel.log file. Look at the second finding, where was the intel info found?

My first question on this task asks me to use the intelligence framework to locate in anomaly.



```
ubuntu@ip-10-10-138-50:~/Desktop/Exercise-Files/TASK-8$ zeek -C -r case1.pcap intelligence-demo.zeek
ubuntu@ip-10-10-138-50:~/Desktop/Exercise-Files/TASK-8$ ls
case1.pcap  clear-logs.sh  conn.log  dhcp.log  dns.log  file-extract-demo.zeek  files.log  hash-demo.zeek  http.log  intel.log  intelligence-demo.zeek  packet_filter.log  pe.log
ubuntu@ip-10-10-138-50:~/Desktop/Exercise-Files/TASK-8$ cat intel.log
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   intel
#open   2023-12-13-22-04-59
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       seen.indicator  seen.indicator_type     seen.where      seen.node       matched sources fuid    file_mime_type  file_desc
#types  time    string  addr    port    addr    port    string  enum    enum    string  set[enum]       set[string]     string  string  string
1561667898.779213       CxfXug13NUaIDb9QPe      10.6.27.102     53770   10.6.27.1       53      smart-fax.com   Intel::DOMAIN   DNS::IN_REQUEST zeek    Intel::DOMAIN   TASK-8-Demo     -       -       -
1561667898.911759       C5Rl2T3Vs244AKPuol      10.6.27.102     49162   107.180.50.162  80      smart-fax.com   Intel::DOMAIN   HTTP::IN_HOST_HEADER    zeek    Intel::DOMAIN   TASK-8-Demo     -       -       -
```

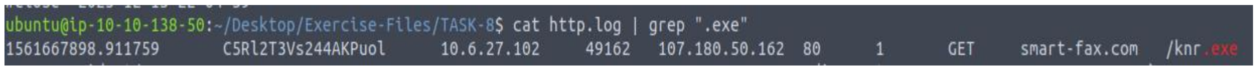
The framework gives me a lot of useful information like the domain, where the ports used, and where the anomaly was located. But since I'm only looking for the information, I can cut it down.



```
untu@ip-10-10-138-50:~/Desktop/Exercise-Files/TASK-8$ cat intel.log | zeek-cut seen.where
S::IN_REQUEST
HTTP::IN_HOST_HEADER
```

Looks like one anomaly was found in a DNS request and the other in a HTTP host header.

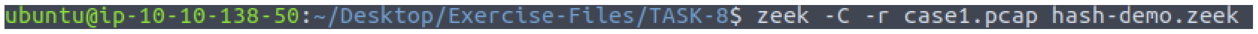
Question 2:
Investigate the http.log file. What is the name of the downloaded .exe file?



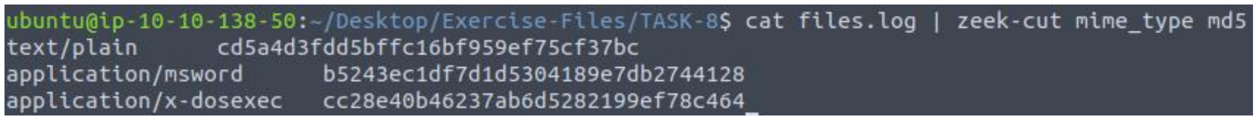
```
ubuntu@ip-10-10-138-50:~/Desktop/Exercise-Files/TASK-8$ cat http.log | grep ".exe"
1561667898.911759       C5Rl2T3Vs244AKPuol      10.6.27.102     49162   107.180.50.162  80      1       GET     smart-fax.com   /knr.exe
```

Question 3:
Investigate the case1.pcap file with hash-demo.zeek script. Investigate the files.log file. What is the MD5 hash of the downloaded .exe file?

Now I am looking for the MD5 hash value of the given exe using the hash framework.



```
ubuntu@ip-10-10-138-50:~/Desktop/Exercise-Files/TASK-8$ zeek -C -r case1.pcap hash-demo.zeek
```

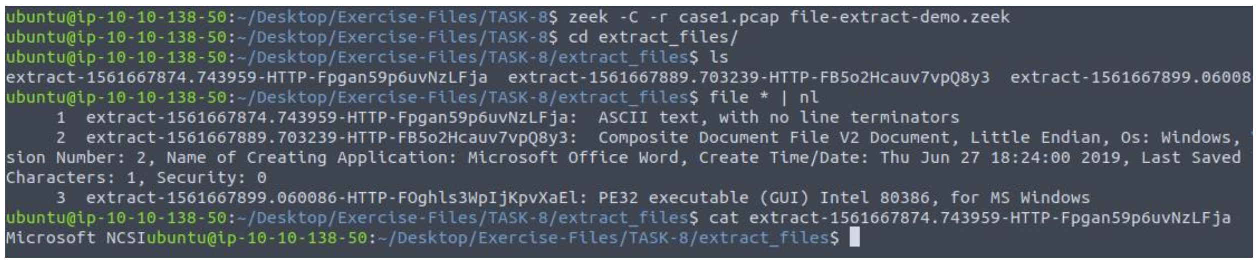


```
ubuntu@ip-10-10-138-50:~/Desktop/Exercise-Files/TASK-8$ cat files.log | zeek-cut mime_type md5
text/plain      cd5a4d3fdd5bffc16bf959ef75cf37bc
application/msword      b5243ec1df7d1d5304189e7db2744128
application/x-dosexec   cc28e40b46237ab6d5282199ef78c464
```

This framework can be used in many ways for intrusion prevention by locating potential anomalies and being able to check this with a given hash value.

Question 4:

Investigate the case1.pcap file with file-extract-demo.zeek script. Investigate the "extract_files" folder. Review the contents of the text file. What is written in the file?

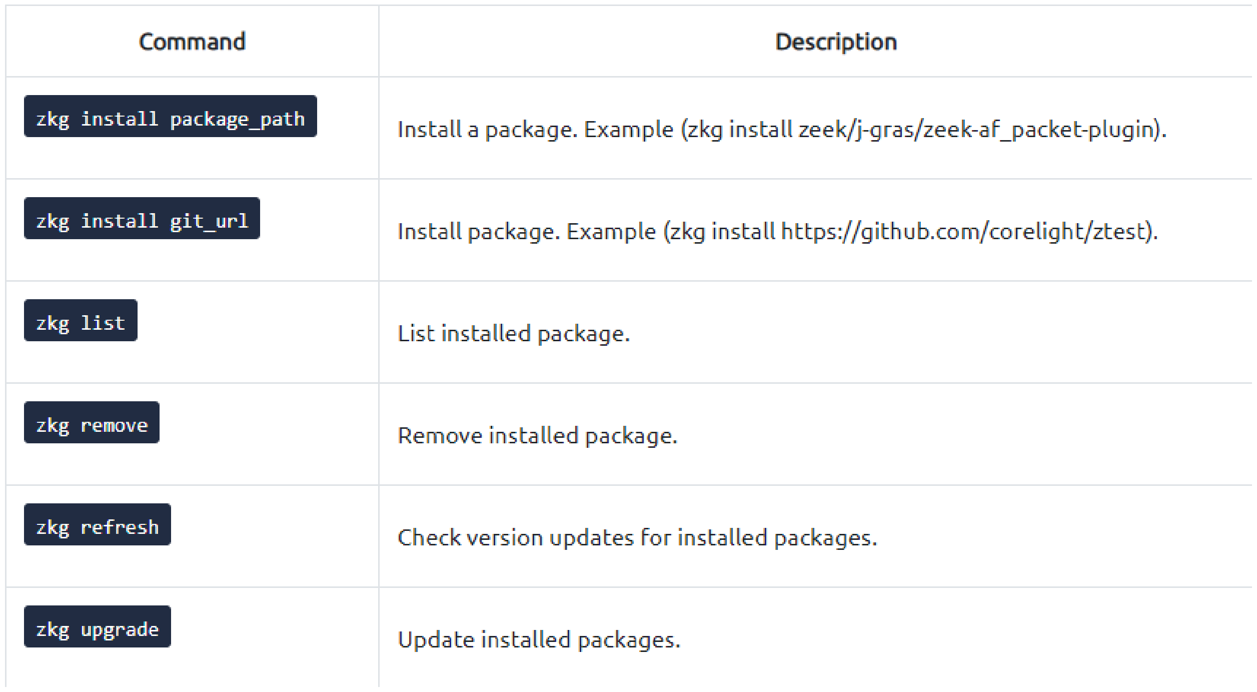


```
ubuntu@ip-10-10-138-50:~/Desktop/Exercise-Files/TASK-8$ zeek -C -r case1.pcap file-extract-demo.zeek
ubuntu@ip-10-10-138-50:~/Desktop/Exercise-Files/TASK-8$ cd extract_files/
ubuntu@ip-10-10-138-50:~/Desktop/Exercise-Files/TASK-8/extract_files$ ls
extract-1561667874.743959-HTTP-Fpgan59p6uvNzLFja  extract-1561667889.703239-HTTP-FB5o2Hcauv7vpQ8y3  extract-1561667899.06008
ubuntu@ip-10-10-138-50:~/Desktop/Exercise-Files/TASK-8/extract_files$ file * | nl
     1  extract-1561667874.743959-HTTP-Fpgan59p6uvNzLFja:  ASCII text, with no line terminators
     2  extract-1561667889.703239-HTTP-FB5o2Hcauv7vpQ8y3:  Composite Document File V2 Document, Little Endian, Os: Windows,
sion Number: 2, Name of Creating Application: Microsoft Office Word, Create Time/Date: Thu Jun 27 18:24:00 2019, Last Saved
Characters: 1, Security: 0
     3  extract-1561667899.060086-HTTP-FOghls3WpIjKpvXaEl: PE32 executable (GUI) Intel 80386, for MS Windows
ubuntu@ip-10-10-138-50:~/Desktop/Exercise-Files/TASK-8/extract_files$ cat extract-1561667874.743959-HTTP-Fpgan59p6uvNzLFja
Microsoft NCSIubuntu@ip-10-10-138-50:~/Desktop/Exercise-Files/TASK-8/extract_files$
```

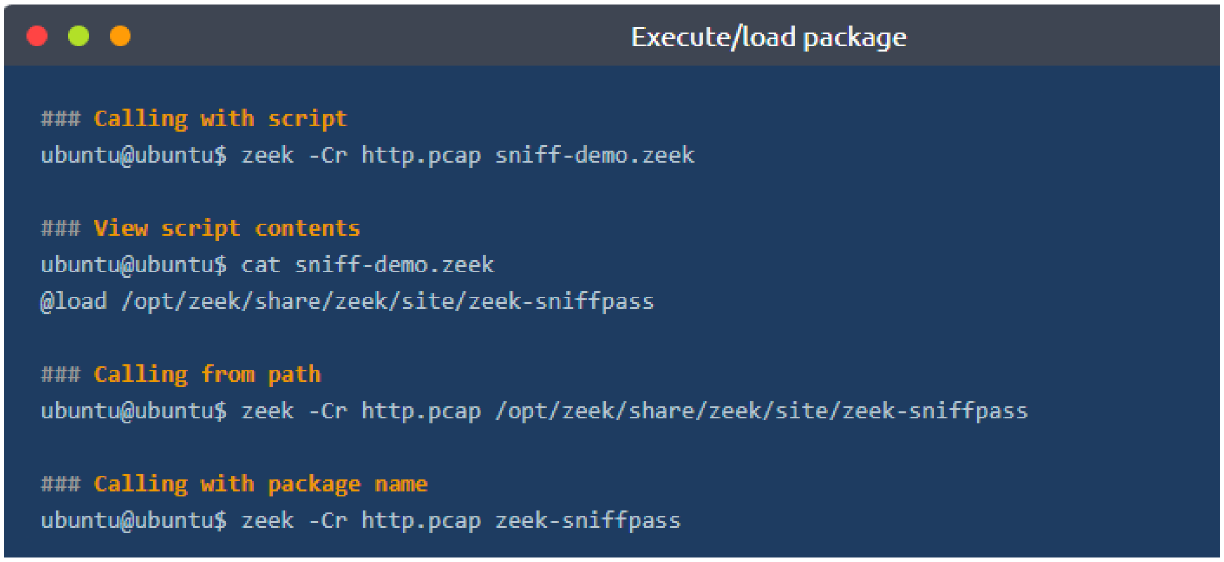
The first line I extract the files using the framework.
Then moving into the folder created, I can use *file * | nl* to locate which of the three files are a text file.
Finally, using cat to view the contents of the file I find my answer.

Task 4: Packages

Zeek comes with a package manager function that allows users to install third-party scripts to extend Zeek's functionalities. Using the *zkg* command, I can install, load, remove and even create packages.

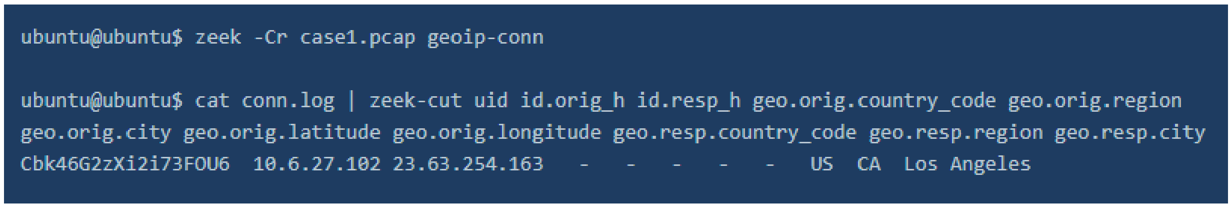


Command	Description
`zkg install package_path`	Install a package. Example (zkg install zeek/j-gras/zeek-af_packet-plugin).
`zkg install git_url`	Install package. Example (zkg install https://github.com/corelight/ztest).
`zkg list`	List installed package.
`zkg remove`	Remove installed package.
`zkg refresh`	Check version updates for installed packages.
`zkg upgrade`	Update installed packages.



```
### Calling with script
ubuntu@ubuntu$ zeek -Cr http.pcap sniff-demo.zeek

### View script contents
ubuntu@ubuntu$ cat sniff-demo.zeek
@load /opt/zeek/share/zeek/site/zeek-sniffpass

### Calling from path
ubuntu@ubuntu$ zeek -Cr http.pcap /opt/zeek/share/zeek/site/zeek-sniffpass

### Calling with package name
ubuntu@ubuntu$ zeek -Cr http.pcap zeek-sniffpass
```

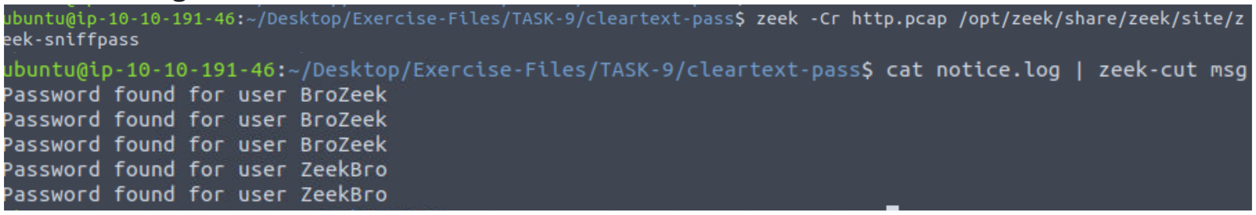
Another example of a useful package called *geoip-conn* can provide geolocation information with an associated IP address.



```
ubuntu@ubuntu$ zeek -Cr case1.pcap geoip-conn

ubuntu@ubuntu$ cat conn.log | zeek-cut uid id.orig_h id.resp_h geo.orig.country_code geo.orig.region
geo.orig.city geo.orig.latitude geo.orig.longitude geo.resp.country_code geo.resp.region geo.resp.city
Cbk46G2zXi2i73FOU6  10.6.27.102 23.63.254.163   -   -   -   -   -   US  CA  Los Angeles
```

Calling on a package uses the same syntax as a other Zeek commands. Make sure to specify the file location if not in the same directory.

Question 1:
Investigate the http.pcap file with the zeek-sniffpass module. Investigate the notice.log file. Which username has more module hits?

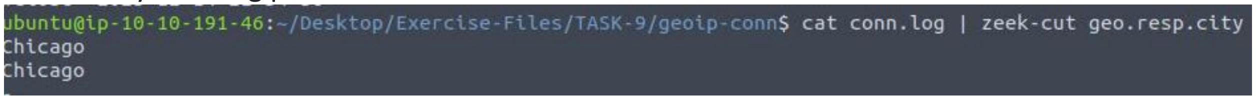


```
ubuntu@ip-10-10-191-46:~/Desktop/Exercise-Files/TASK-9/cleartext-pass$ zeek -Cr http.pcap /opt/zeek/share/zeek/site/z
eek-sniffpass
ubuntu@ip-10-10-191-46:~/Desktop/Exercise-Files/TASK-9/cleartext-pass$ cat notice.log | zeek-cut msg
Password found for user BroZeek
Password found for user BroZeek
Password found for user BroZeek
Password found for user ZeekBro
Password found for user ZeekBro
```

BroZeek is the answer I am looking for

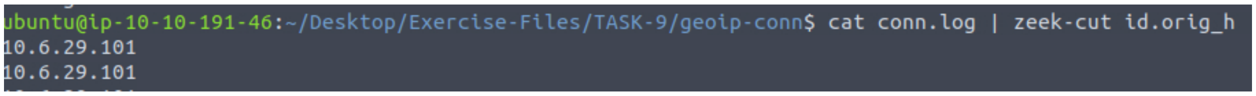
Question 2:
Investigate the case2.pcap file with geoip-conn module. Investigate the conn.log file. What is the name of the identified City?

I used the command *zeek -Cr case2.pcap geoip-conn.zeek* to extract the data I am looking for. Then using the cat command along with *zeek-cut*. I can see exactly where the activity is taking place.



```
ubuntu@ip-10-10-191-46:~/Desktop/Exercise-Files/TASK-9/geoip-conn$ cat conn.log | zeek-cut geo.resp.city
Chicago
Chicago
```

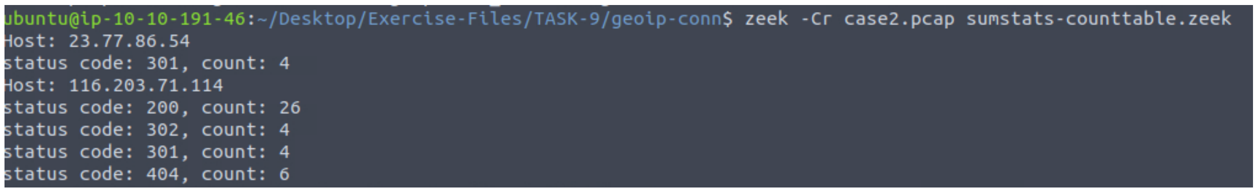
Question 3:
Which IP address is associated with the identified City?



```
ubuntu@ip-10-10-191-46:~/Desktop/Exercise-Files/TASK-9/geoip-conn$ cat conn.log | zeek-cut id.orig_h
10.6.29.101
10.6.29.101
```

Question 4:
Investigate the case2.pcap file with sumstats-counttable.zeek script. How many types of status codes are there in the given traffic capture?

Using the sumstats Zeek script it lists all the status codes it finds and I am able to count the number of unique numbers to find my answer.



```
ubuntu@ip-10-10-191-46:~/Desktop/Exercise-Files/TASK-9/geoip-conn$ zeek -Cr case2.pcap sumstats-counttable.zeek
Host: 23.77.86.54
status code: 301, count: 4
Host: 116.203.71.114
status code: 200, count: 26
status code: 302, count: 4
status code: 301, count: 4
status code: 404, count: 6
```