INTRUSION DETECTION SYSTEM

Zeek Cluster
Scenarios

Submitted By
Leen Falah
Dana Abu Shaweesh

1 Abstract

In the ever-evolving landscape of cybersecurity, the ability to dissect and understand the complex choreography of network traffic is paramount. As malicious actors continuously refine their tactics, defenders need advanced tools to unravel the intricacies of these dances. This report explores the multifaceted functionalities of Zeek, a versatile network security monitoring tool, with a particular emphasis on its anomaly-based detection capabilities.

2 Introduction

Zeek, formerly known as Bro, is a powerful network traffic analyzer. Imagine it as a watchful owl, perched high on a branch, observing the intricate dance of digital communication. It dissects protocols with the precision of a surgeon, extracting valuable insights from the raw data that courses through its veins. But its purpose is not mere observation; it is to expose the malicious intent hidden within, unmasking the whispers of potential attacks before they can cause harm.

In the complex realm of cybersecurity, the effectiveness of network security monitoring tools plays a pivotal role in fortifying digital landscapes against evolving threats. At the forefront of these tools is Zeek, This report embarks on an in-depth exploration of Zeek's multifaceted functionalities, delving into its adeptness in deciphering the intricate choreography of network traffic across crucial protocols such as SSH, HTTP, DNS, and UDP and threats like SQL Injection,Buffer Overflow,Brute-Force SSH.

As cyber threats become increasingly sophisticated, traditional security mechanisms often fall short in identifying subtle deviations from the norm. Herein lies the strength of Zeek's anomaly-based detection approach, where it excels in uncovering irregular patterns and behaviors within network traffic that may signify potential security breaches.

3 Buffer Overflow

It's a type of cyberattack that exploits a program's vulnerability to overwrite adjacent memory locations with malicious code. It occurs when more data is written to a buffer (a temporary data storage area) than it's designed to hold. This overflow can corrupt, crash, or even hijack control of the program, leading to serious consequences.

Increasing Buffer Sizes in Python Code as shown in the figure below:


```python
#!/usr/bin/python3

import socket
import sys
from time import sleep

buffer = b'A' * 1000  # Increase buffer size, using bytes

while True:
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.settimeout(2)
        s.connect(('192.168.1.13', 80))

        # Send a simple HTTP request
        http_request = b"GET / HTTP/1.1\r\nHost: 192.168.1.13\r\n\r\n"
        s.send(http_request)

        print('[*] Sending buffer with length: ' + str(len(buffer)))
        s.send(buffer)
        s.close()
        sleep(2)
        buffer = buffer + b'A' * 1000

    except:
        print('[*] Crash occurred at buffer length: ' + str(len(buffer) - 1000))
        sys.exit()
```

Figure 1: Buffer Overflow code

Run the code to start buffer overflow attack to victim machine as shown in the figure below:

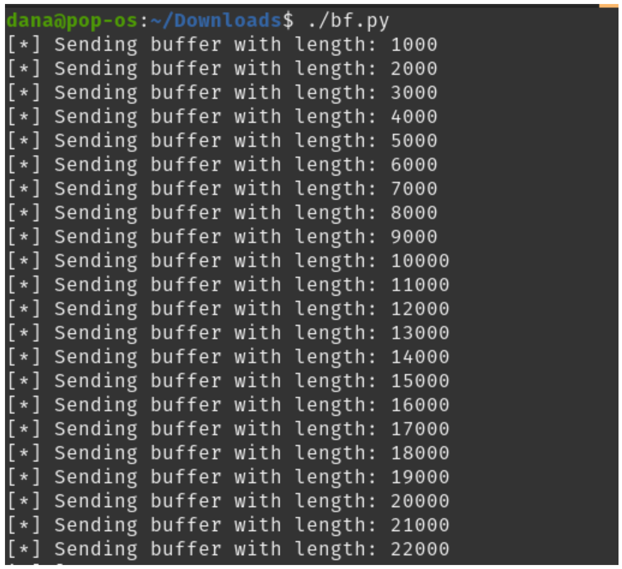


Figure 2: Run Buffer Overflow code

Bro (Zeek) can be used to detect buffer overflow attempts using anomaly-based techniques within a cluster environment as shown in the figure below:

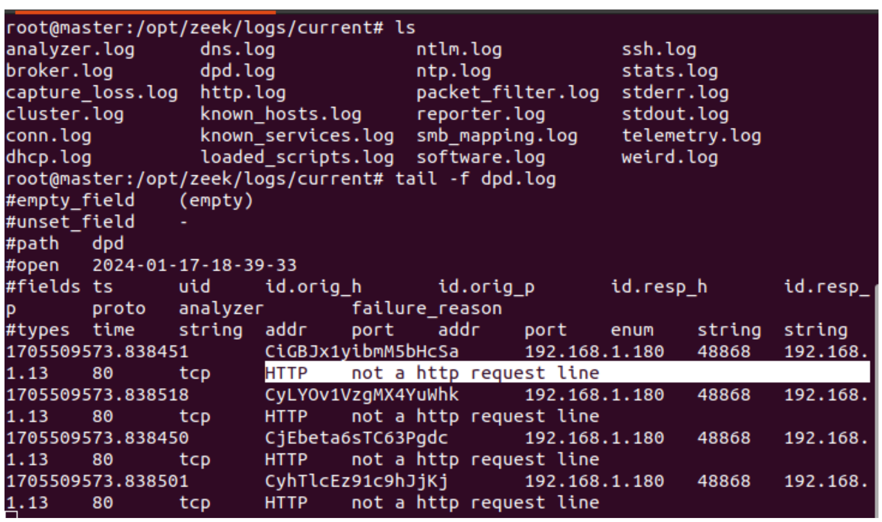


Figure 3: Detect Buffer Overflow by Zeek

4 SQL Injection

SQL injection is a type of cyber attack that occurs when an attacker is able to manipulate a database query by injecting malicious SQL (Structured Query Language) code into the input fields of a vulnerable application.

Start SQL Injection attempt from attacker machine

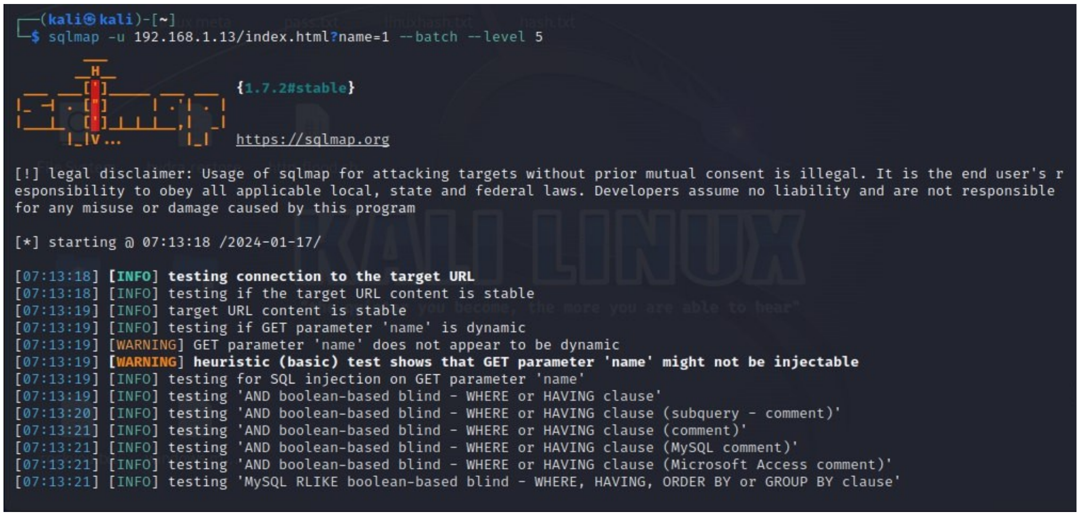


Figure 4: SQL Injection command

Zeek can detect SQL injection attempts by analyzing network traffic and identifying patterns or anomalies in the SQL-related activity as shown in the figures below :

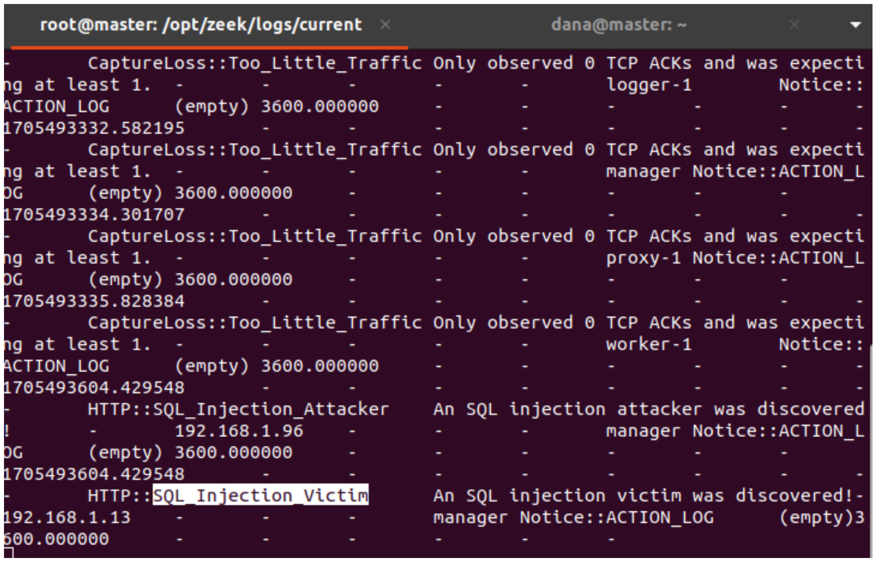


Figure 5: Detect SQL Injection attempt from by zeek

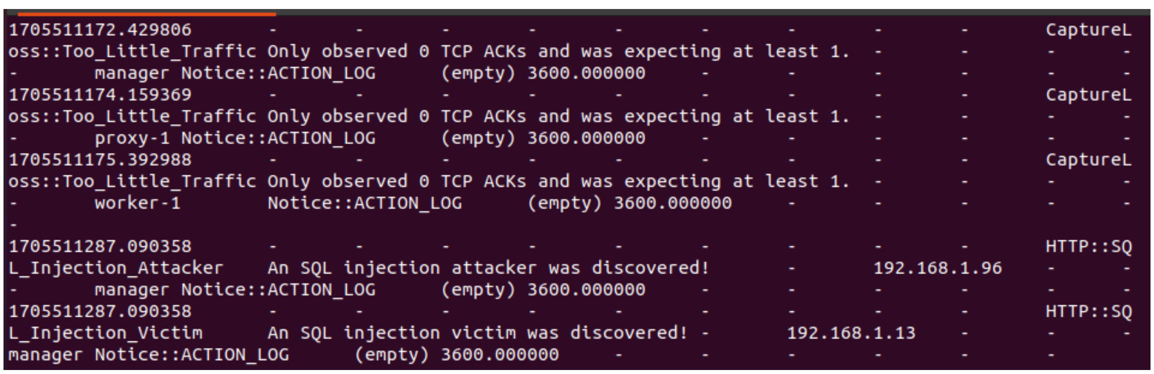
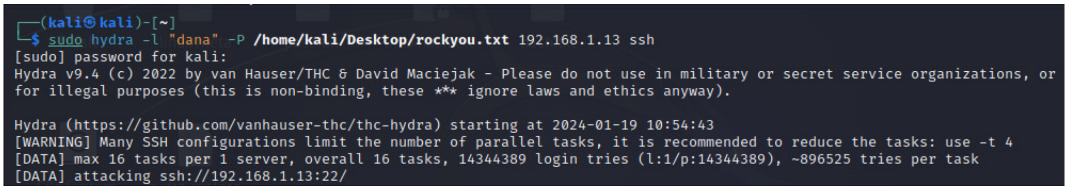


Figure 6: Detect SQL Injection attempt from by zeek

5 SSH Bruteforce

SSH (Secure Shell) brute force attacks are attempts by malicious actors to gain unauthorized access to a system by systematically trying a large number of usernames and passwords.

Start guessing passwords by using hydra tool as shown in the figure below :



```
┌──(kali㉿kali)-[~]
└─$ sudo hydra -l "dana" -P /home/kali/Desktop/rockyou.txt 192.168.1.13 ssh
[sudo] password for kali:
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or
for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-01-19 10:54:43
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344389 login tries (l:1/p:14344389), ~896525 tries per task
[DATA] attacking ssh://192.168.1.13:22/
```

Figure 7: hydra command to guess passwords

Zeek Detect Guessing SSH Password As Shown In The Figure Below :

```
root@master:/opt/zeek/logs/current# tail -f notice.log
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   notice
#open   2024-01-18-21-01-13
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       fuid    file_mim
e_type  file_desc       proto   note    msg     sub     src     dst     p       n       peer_descr      a
ctions  email_dest      suppress_for    remote_location.country_code    remote_location.region  remote_l
ocation.city    remote_location.latitude        remote_location.longitude       community_id
#types  time    string  addr    port    addr    port    string  string  string  enum    enum    string s
tring   addr    addr    port    count   string  set[enum]       set[string]     interval        string s
tring   string  double  double  string
1705604471.661249       -       -       -       -       -       -       -       -       -       SSH::Pas
sword_Guessing  192.168.1.96 appears to be guessing SSH passwords (seen in 30 connections).     Sampled
servers:  192.168.1.13, 192.168.1.13, 192.168.1.13, 192.168.1.13, 192.168.1.13  192.168.1.96    -       -
-       manager Notice::ACTION_LOG      (empty) 3600.000000     -       -       -       -       -       -
```

Figure 8: Detect guessing ssh password by zeek

6 HTTP Traffic

Zeek is a powerful network traffic analysis tool that can monitor and analyze network traffic in real-time. It can be used to identify a wide range of malicious activity, including HTTP attacks.

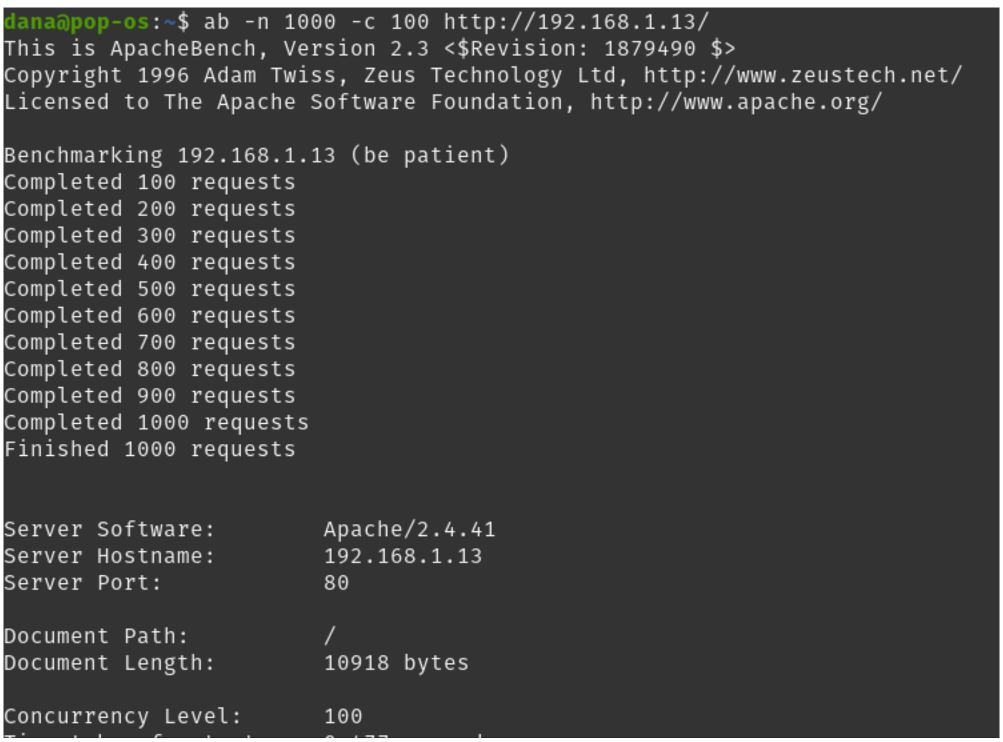
The top line in the figure below shows the command used to run ApacheBench: **ab -n 1000 c 100 http://192.168.1.13/.** This tells us that ApacheBench was run with the following options:

-n 1000: This specifies the number of requests to make (1000 in this case).

-c 100: This specifies the concurrency level, meaning the number of concurrent requests to make (100 in this case).

http://192.168.1.13/: This specifies the target URL to benchmark.

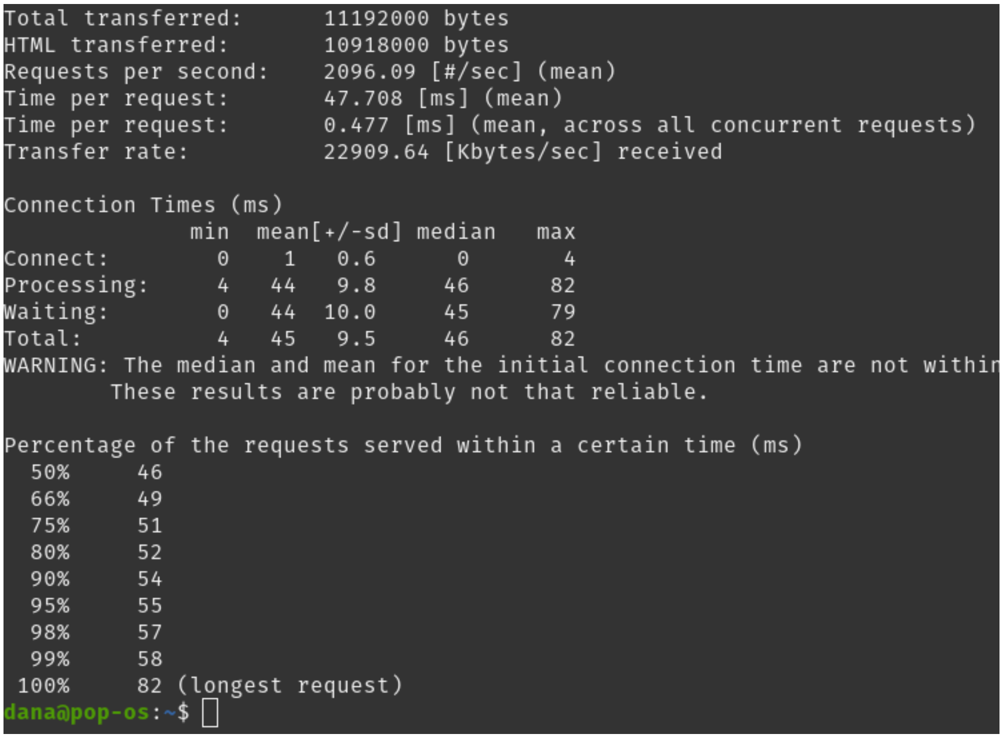
Once the benchmark is finished, it displays information about the server, including its software **(Server Software: Apache/2.4.41), hostname (Server Hostname: 192.168.1.13), port (Server Port: 80), document path (Document Path: /), document length (Document Length: 10918 bytes), and** concurrency level (Concurrency Level: 100).



```
dana@pop-os:~$ ab -n 1000 -c 100 http://192.168.1.13/
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.1.13 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:        Apache/2.4.41
Server Hostname:        192.168.1.13
Server Port:            80

Document Path:          /
Document Length:        10918 bytes

Concurrency Level:      100
```

Figure 9: HTTP requests (Attacker machine)

the images from first machine to second machine shows the results of a performance test conducted on a web server using ApacheBench. It provides information about the server's response time and performance under a specific load as shown in the figure below.



```
Total transferred:      11192000 bytes
HTML transferred:       10918000 bytes
Requests per second:    2096.09 [#/sec] (mean)
Time per request:       47.708 [ms] (mean)
Time per request:       0.477 [ms] (mean, across all concurrent requests)
Transfer rate:          22909.64 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    1   0.6      0       4
Processing:     4   44   9.8     46      82
Waiting:        0   44  10.0     45      79
Total:          4   45   9.5     46      82
WARNING: The median and mean for the initial connection time are not withir
        These results are probably not that reliable.

Percentage of the requests served within a certain time (ms)
  50%     46
  66%     49
  75%     51
  80%     52
  90%     54
  95%     55
  98%     57
  99%     58
 100%     82 (longest request)
dana@pop-os:~$
```

Figure 10: HTTP requests (Attacker machine)

The command in the figure below **sudo tail -f /opt/zeek/logs/current/http.log**, is used to view the Zeek HTTP log file in real-time.

This command in second machine (victim) , when applied, shows the traffic information captured by Zeek and stored in a log file , such as victim ip and attacker ip and port number ... etc.

By default, Zeek stores its logs in the /opt/zeek/logs/current directory, with separate files for different protocols like HTTP, DNS, and SSH.

Figure 11: HTTP response (Victim machine)

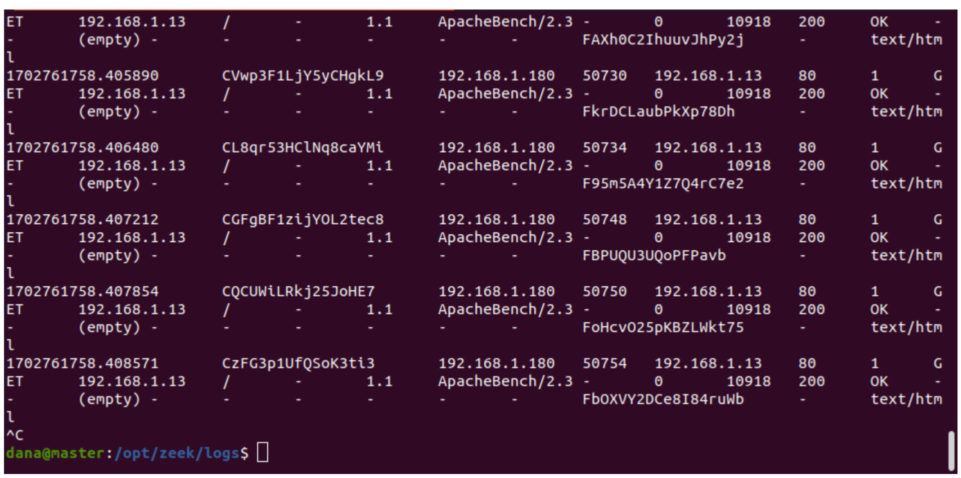
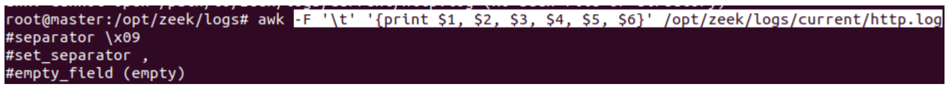
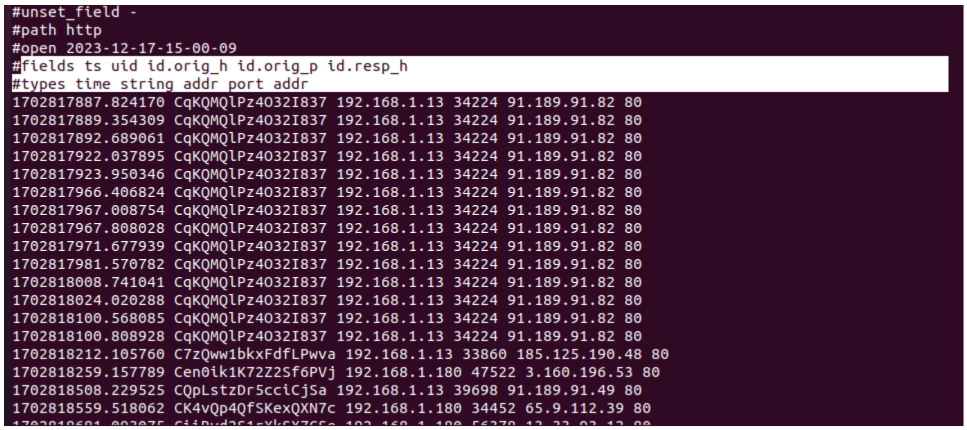


Figure 12: HTTP response (Victim machine)



```
root@master:/opt/zeek/logs# awk -F '\t' '{print $1, $2, $3, $4, $5, $6}' /opt/zeek/logs/current/http.log
#separator \x09
#set_separator ,
#empty_field (empty)
```

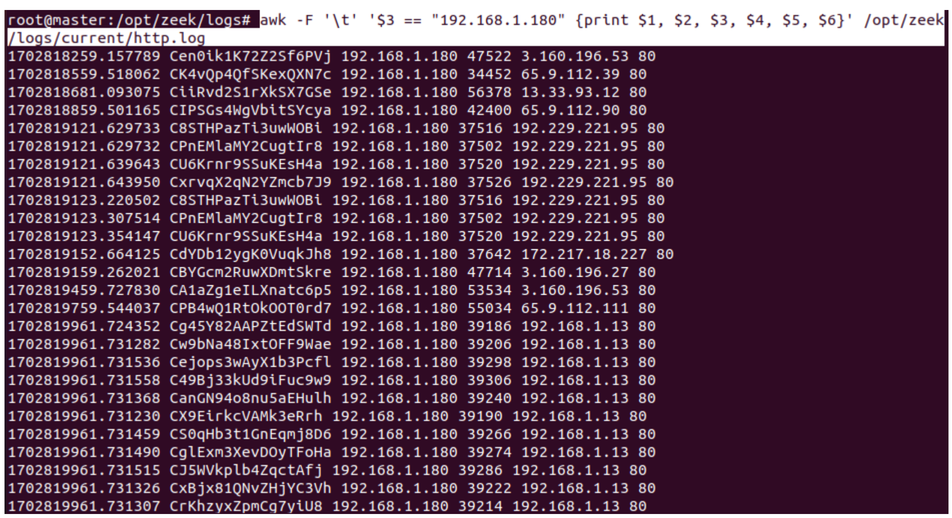
Figure 13: HTTP response (Victim machine)



```
#unset_field -
#path http
#open 2023-12-17-15-00-09
#fields ts uid id.orig_h id.orig_p id.resp_h
#types time string addr port addr
1702817887.824170 CqKQMQlPz4O32I837 192.168.1.13 34224 91.189.91.82 80
1702817889.354309 CqKQMQlPz4O32I837 192.168.1.13 34224 91.189.91.82 80
1702817892.689061 CqKQMQlPz4O32I837 192.168.1.13 34224 91.189.91.82 80
1702817922.037895 CqKQMQlPz4O32I837 192.168.1.13 34224 91.189.91.82 80
1702817923.950346 CqKQMQlPz4O32I837 192.168.1.13 34224 91.189.91.82 80
1702817966.406824 CqKQMQlPz4O32I837 192.168.1.13 34224 91.189.91.82 80
1702817967.008754 CqKQMQlPz4O32I837 192.168.1.13 34224 91.189.91.82 80
1702817967.808028 CqKQMQlPz4O32I837 192.168.1.13 34224 91.189.91.82 80
1702817971.677939 CqKQMQlPz4O32I837 192.168.1.13 34224 91.189.91.82 80
1702817981.570782 CqKQMQlPz4O32I837 192.168.1.13 34224 91.189.91.82 80
1702818008.741041 CqKQMQlPz4O32I837 192.168.1.13 34224 91.189.91.82 80
1702818024.020288 CqKQMQlPz4O32I837 192.168.1.13 34224 91.189.91.82 80
1702818100.568085 CqKQMQlPz4O32I837 192.168.1.13 34224 91.189.91.82 80
1702818100.808928 CqKQMQlPz4O32I837 192.168.1.13 34224 91.189.91.82 80
1702818212.105760 C7zQww1bkxFdfLPwva 192.168.1.13 33860 185.125.190.48 80
1702818259.157789 Cen0ik1K72Z2Sf6PVj 192.168.1.180 47522 3.160.196.53 80
1702818508.229525 CQpLstzDr5cciCjSa 192.168.1.13 39698 91.189.91.49 80
1702818559.518062 CK4vQp4QfSKexQXN7c 192.168.1.180 34452 65.9.112.39 80
```

Figure 14: HTTP response (Victim machine)

This command is used to filter captured traffic so that it appears in clear, organized fields



```
root@master:/opt/zeek/logs# awk -F '\t' '$3 == "192.168.1.180" {print $1, $2, $3, $4, $5, $6}' /opt/zeek/logs/current/http.log
1702818259.157789 Cen0ik1K72Z2Sf6PVj 192.168.1.180 47522 3.160.196.53 80
1702818559.518062 CK4vQp4QfSKexQXN7c 192.168.1.180 34452 65.9.112.39 80
1702818681.093075 CiiRvd2S1rXkSX7GSe 192.168.1.180 56378 13.33.93.12 80
1702818859.501165 CIPSGs4WgVbitSYcya 192.168.1.180 42400 65.9.112.90 80
1702819121.629733 C8STHPazTi3uwWOBi 192.168.1.180 37516 192.229.221.95 80
1702819121.629732 CPnEMlaMY2CugtIr8 192.168.1.180 37502 192.229.221.95 80
1702819121.639643 CU6Krnr9SSuKEsH4a 192.168.1.180 37520 192.229.221.95 80
1702819121.643950 CxrvqX2qN2YZmcb7J9 192.168.1.180 37526 192.229.221.95 80
1702819123.220502 C8STHPazTi3uwWOBi 192.168.1.180 37516 192.229.221.95 80
1702819123.307514 CPnEMlaMY2CugtIr8 192.168.1.180 37502 192.229.221.95 80
1702819123.354147 CU6Krnr9SSuKEsH4a 192.168.1.180 37520 192.229.221.95 80
1702819152.664125 CdYDb12ygK0VuqkJh8 192.168.1.180 37642 172.217.18.227 80
1702819159.262021 CBYGcm2RuwXDmtSkre 192.168.1.180 47714 3.160.196.27 80
1702819459.727830 CA1aZg1eILXnatc6p5 192.168.1.180 53534 3.160.196.53 80
1702819759.544037 CPB4wQ1RtOkOOT0rd7 192.168.1.180 55034 65.9.112.111 80
1702819961.724352 Cg45Y82AAPZtEdSWTd 192.168.1.180 39186 192.168.1.13 80
1702819961.731282 Cw9bNa48IxtOFF9Wae 192.168.1.180 39206 192.168.1.13 80
1702819961.731536 Cejops3wAyX1b3Pcfl 192.168.1.180 39298 192.168.1.13 80
1702819961.731558 C49Bj33kUd9iFuc9w9 192.168.1.180 39306 192.168.1.13 80
1702819961.731368 CanGN94o8nu5aEHulh 192.168.1.180 39240 192.168.1.13 80
1702819961.731230 CX9EirkcVAMk3eRrh 192.168.1.180 39190 192.168.1.13 80
1702819961.731459 CS0qHb3t1GnEqmj8D6 192.168.1.180 39266 192.168.1.13 80
1702819961.731490 CglExm3XevDOyTFoHa 192.168.1.180 39274 192.168.1.13 80
1702819961.731515 CJ5WVkplb4ZqctAfj 192.168.1.180 39286 192.168.1.13 80
1702819961.731326 CxBjx81QNvZHjYC3Vh 192.168.1.180 39222 192.168.1.13 80
1702819961.731307 CrKhzyxZpmCg7yiU8 192.168.1.180 39214 192.168.1.13 80
```

Figure 15: HTTP response (Victim machine)

This command is used to filter captured traffic from Attacker machine so that it appears Attacker IP address and some information .

Zeek is successfully detecting and logging the suspicious activity, to analyze the attack and take appropriate security measures.

7 DNS Queries

Zeek acts like a vigilant detective in the digital realm, watching the network with keen eyes. When it sees suspicious DNS traffic, like unusual domain requests or sudden spikes in activity, it raises alarms. It analyzes patterns, compares them to past threats, and identifies potential DNS attacks hiding in the shadows.

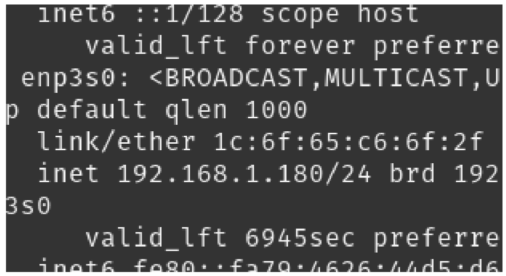


Figure 16: Attacker IP address

The command **nslookup zajel.najah.edu** at attacker machine performs a DNS lookup of the domain zajel.najah.edu as shown in the figure below.

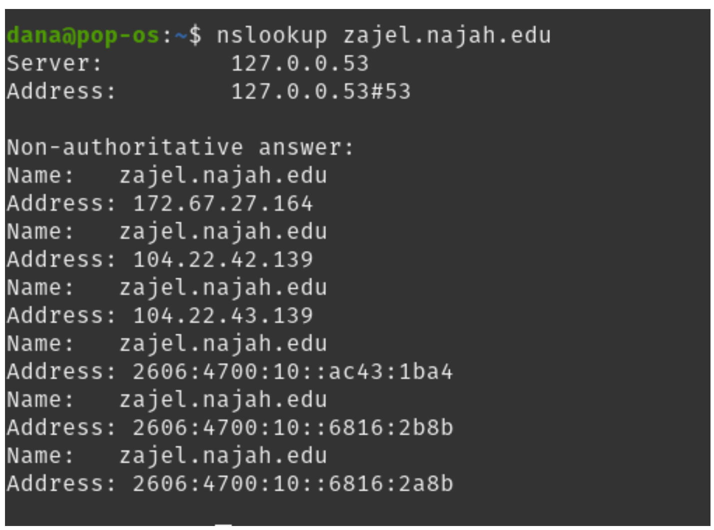
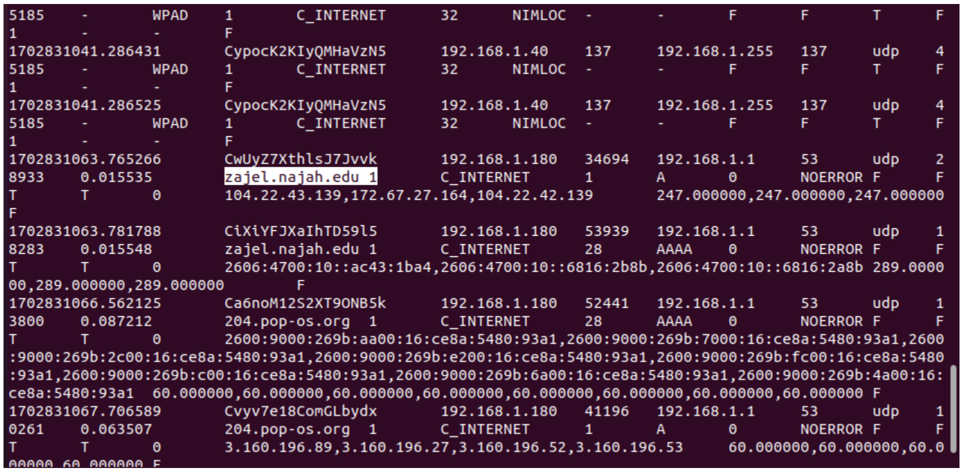


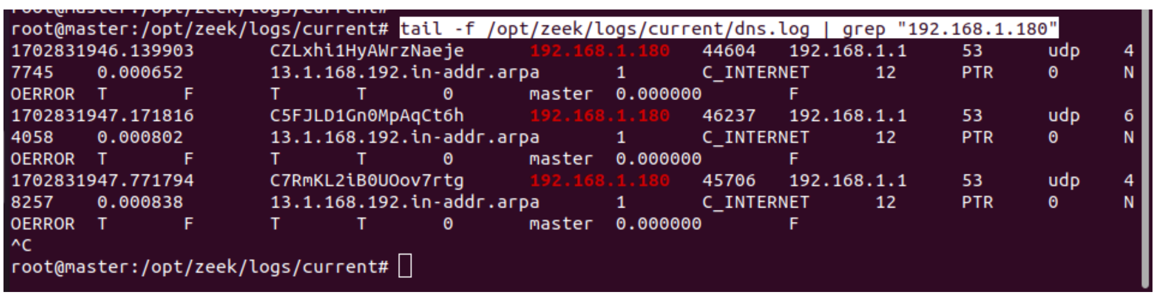
Figure 17: nslookup command (Attacker machine)

Detecting the DNS Lookup: Zeek monitors outgoing DNS traffic from the attacker's machine. and capture the corresponding DNS request and response messages as shown in the figure below.



```
5185    -       WPAD    1       C_INTERNET      32      NIMLOC  -       -       F       F       T       F
1       -       -       F
1702831041.286431       CypocK2KIyQMHaVzN5      192.168.1.40    137     192.168.1.255   137     udp     4
5185    -       WPAD    1       C_INTERNET      32      NIMLOC  -       -       F       F       T       F
1       -       -       F
1702831041.286525       CypocK2KIyQMHaVzN5      192.168.1.40    137     192.168.1.255   137     udp     4
5185    -       WPAD    1       C_INTERNET      32      NIMLOC  -       -       F       F       T       F
1       -       -       F
1702831063.765266       CwUyZ7XthlsJ7Jvvk       192.168.1.180   34694   192.168.1.1     53      udp     2
8933    0.015535        zajel.najah.edu 1       C_INTERNET      1       A       0       NOERROR F       F
T       T       0       104.22.43.139,172.67.27.164,104.22.42.139       247.000000,247.000000,247.000000
F
1702831063.781788       CiXiYFJXaIhTD59l5       192.168.1.180   53939   192.168.1.1     53      udp     1
8283    0.015548        zajel.najah.edu 1       C_INTERNET      28      AAAA    0       NOERROR F       F
T       T       0       2606:4700:10::ac43:1ba4,2606:4700:10::6816:2b8b,2606:4700:10::6816:2a8b 289.0000
00,289.000000,289.000000        F
1702831066.562125       Ca6noM12S2XT9ONB5k      192.168.1.180   52441   192.168.1.1     53      udp     1
3800    0.087212        204.pop-os.org  1       C_INTERNET      28      AAAA    0       NOERROR F       F
T       T       0       2600:9000:269b:aa00:16:ce8a:5480:93a1,2600:9000:269b:7000:16:ce8a:5480:93a1,2600
:9000:269b:2c00:16:ce8a:5480:93a1,2600:9000:269b:e200:16:ce8a:5480:93a1,2600:9000:269b:fc00:16:ce8a:5480
:93a1,2600:9000:269b:c00:16:ce8a:5480:93a1,2600:9000:269b:6a00:16:ce8a:5480:93a1,2600:9000:269b:4a00:16:
ce8a:5480:93a1  60.000000,60.000000,60.000000,60.000000,60.000000,60.000000,60.000000,60.000000 F
1702831067.706589       Cvyv7e18ComGLbydx       192.168.1.180   41196   192.168.1.1     53      udp     1
0261    0.063507        204.pop-os.org  1       C_INTERNET      1       A       0       NOERROR F       F
T       T       0       3.160.196.89,3.160.196.27,3.160.196.52,3.160.196.53     60.000000,60.000000,60.0
00000,60.000000 F
```

Figure 18: Capture traffic by ZEEK (Victim machine)



```
root@master:/opt/zeek/logs/current# tail -f /opt/zeek/logs/current/dns.log | grep "192.168.1.180"
1702831946.139903       CZLxhi1HyAWrzNaeje      192.168.1.180   44604   192.168.1.1     53      udp     4
7745    0.000652        13.1.168.192.in-addr.arpa       1       C_INTERNET      12      PTR     0       N
OERROR  T       F       T       T       0       master  0.000000        F
1702831947.171816       C5FJLD1Gn0MpAqCt6h      192.168.1.180   46237   192.168.1.1     53      udp     6
4058    0.000802        13.1.168.192.in-addr.arpa       1       C_INTERNET      12      PTR     0       N
OERROR  T       F       T       T       0       master  0.000000        F
1702831947.771794       C7RmKL2iB0UOov7rtg      192.168.1.180   45706   192.168.1.1     53      udp     4
8257    0.000838        13.1.168.192.in-addr.arpa       1       C_INTERNET      12      PTR     0       N
OERROR  T       F       T       T       0       master  0.000000        F
^C
root@master:/opt/zeek/logs/current#
```

Figure 19: Capture traffic by ZEEK (Victim machine)

Here it shows the traffic captured by zeek and stored in the **dns.log file** Coming specifically from the attacker's device.

8 Secure Shell(SSH)

The command in the figure below to establish an SSH connection to a server with the IP address 192.168.1.180. The username for the login is ”dana”,to gain access to systems.

```
dana@master:~$ ssh dana@192.168.1.180
The authenticity of host '192.168.1.180 (192.168.1.180)' can't be established.
ECDSA key fingerprint is SHA256:HaJ2g989wxNT7ooFyy+1OAcl8r37DSZ8iVj0LX9D9/U.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.180' (ECDSA) to the list of known hosts.
dana@192.168.1.180's password:
Welcome to Pop!_OS 22.04 LTS (GNU/Linux 6.0.12-76060006-generic x86_64)

 * Homepage: https://pop.system76.com
 * Support:  https://support.system76.com

Last login: Sat Dec 16 18:42:04 2023 from 192.168.1.96
dana@pop-os:~$ exit
logout
Connection to 192.168.1.180 closed.
dana@master:~$ ssh dana@192.168.1.180
dana@192.168.1.180's password:
Welcome to Pop!_OS 22.04 LTS (GNU/Linux 6.0.12-76060006-generic x86_64)

 * Homepage: https://pop.system76.com
 * Support:  https://support.system76.com

Last login: Sun Dec 17 16:47:15 2023 from 192.168.1.13
dana@pop-os:~$
```

Figure 20: Start SSH connection to 192.168.1.180

ls command used to list the contents of the **/opt/zeek/logs/current** directory on a Zeek server,as show in the figure below.

```
root@master:/opt/zeek/logs/current# ls
analyzer.log       files.log            loaded_scripts.log   packet_filter.log   stderr.log
capture_loss.log   http.log             notice.log           software.log        stdout.log
conn.log           known_hosts.log      ntp.log              ssl.log             telemetry.log
dns.log            known_services.log   ocsp.log             stats.log           weird.log
```

Figure 21: List log files by using ls command

tail -f /opt/zeek/logs/current/software.log command on a Zeek server. This command continuously displays the last lines of the /opt/zeek/logs/current/software.log file, which is a log file used by Zeek to record information about the software itself.

```
root@master:/opt/zeek/logs/current# tail -f /opt/zeek/logs/current/software.log
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   software
#open   2023-12-17-18-13-23
#fields ts      host    host_p  software_type   name    version.major   version.minor   version.minor2 v
ersion.minor3   version.addl    unparsed_version
#types  time    addr    port    enum    string  count   count   count   count   string  string
1702829603.717167       192.168.1.180   -       HTTP::BROWSER   Firefox 118     0       -       -       -
Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/118.0
1702829958.152975       192.168.1.180   22      SSH::SERVER     OpenSSH 8       9       -       -       p
1       OpenSSH_8.9p1 Ubuntu-3ubuntu0.1
^C
root@master:/opt/zeek/logs/current#
```

Figure 22: The tail command to shows the last few lines of the software log file

The first line shows the header information for the log file, including the timestamp, unparsed version, and additional version details.
The subsequent lines show individual log entries. Each entry includes the following information:
Timestamp: The date and time the event occurred.
Host: The hostname or IP address of the machine that generated the event(192.168.1.180).
Software type: The type of software involved in the event (OpenSSH server).
Name: The specific name of the software (Firefox web browser).
Version information: (version 118.0).

The figure below shows command **cat /opt/zeek/logs/current/weird.log — grep "192.168.1.180".** This command uses two parts: **cat /opt/zeek/logs/current/weird.log**: This reads the contents of the **/opt/zeek/logs/current/weird.log file** on the system. This file, as its name suggests, contains logs of unusual events detected by Zeek.
— **grep "192.168.1.180"**: This pipes the output of the cat command (the contents of the log file) to the grep command, which filters the lines based on a pattern. In this case, the pattern is the IP address "192.168.1.180". So, this part of the command only shows lines in the log file that mention this specific IP address.

```
root@master:/opt/zeek/logs/current# cat /opt/zeek/logs/current/weird.log | grep "192.168.1.180"
1702824597.235502       CkXidUJh22X6dPMfk       192.168.1.180   39650   18.161.111.75   443     bad_TCP_
checksum        -       F       zeek    TCP
1702825075.536222       CBV8qQ3fZQKV0iqv57      192.168.1.13    39698   192.168.1.180   22      active_c
onnection_reuse -       F       zeek    TCP
root@master:/opt/zeek/logs/current#
```

Figure 23: List weird.log content Regarding a specific IP (192.168.1.180)

Hydra is a password cracking tool that can be used to try to guess passwords for a variety of different services, including SSH. It works by trying a large number of different passwords until it finds one that works.

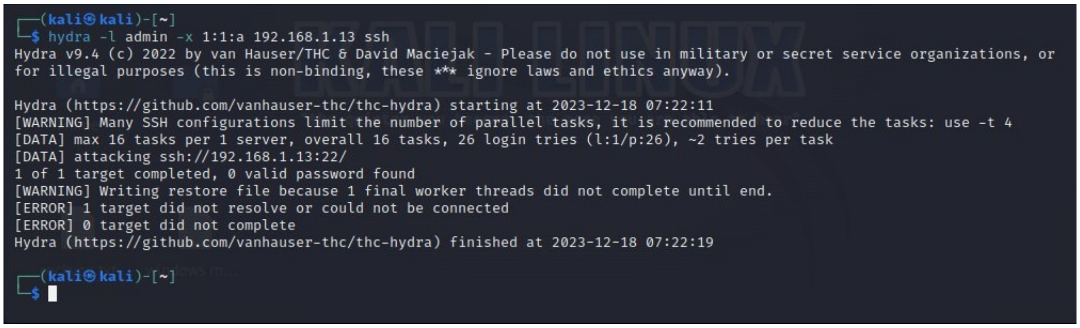


Figure 24: Start Hydra tool to guess the password for an SSH server on the IP address 192.168.1.13

In the figure below search for lines containing the **IP"192.168.1.96"** within the log of SSH connections.
ssh.log file records information about individual SSH connections.

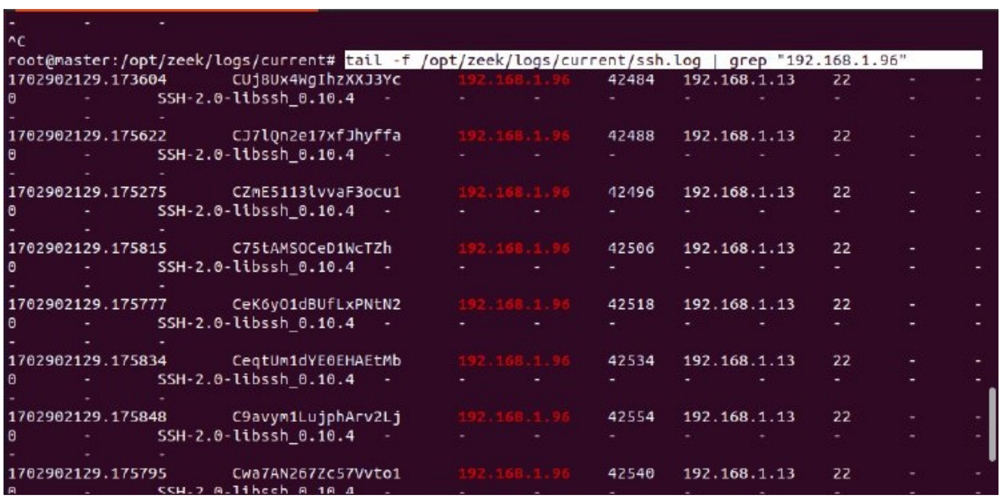


Figure 25: Show the result of ssh.log file especially from 192.168.1.96

9 UDP Port Scanning

The local.zeek file is used to load additional scripts and customizations specific to Zeek deployment.
These customizations can include :Loading additional Zeek policy scripts,Defining local network information,Tuning Zeek settings.
@load directives instruct Zeek to load specific scripts during startup like:
protocols/ssh/detect-bruteforcing: This script detects brute-force login attempts against SSH servers.
protocols/ssh/interesting-hostnames: This script flags login attempts originating from hosts with unusual names, potentially indicating suspicious activity.
protocols/ssh: This script provides general analysis and logging for SSH traffic.
protocols/http/detect-sqli: This script detects potential SQL injection attacks in HTTP traffic.
UDP Scan: This script likely detects and analyzes UDP scans on the network as shown below.

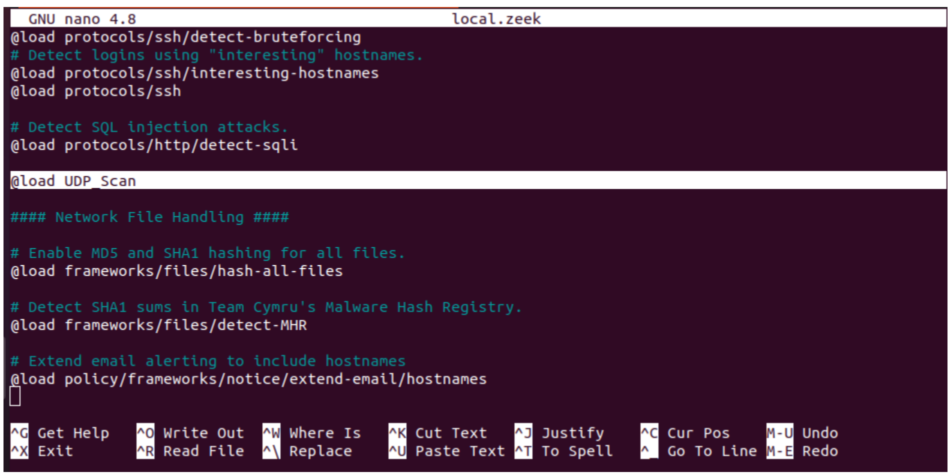


Figure 26: Local.zeek

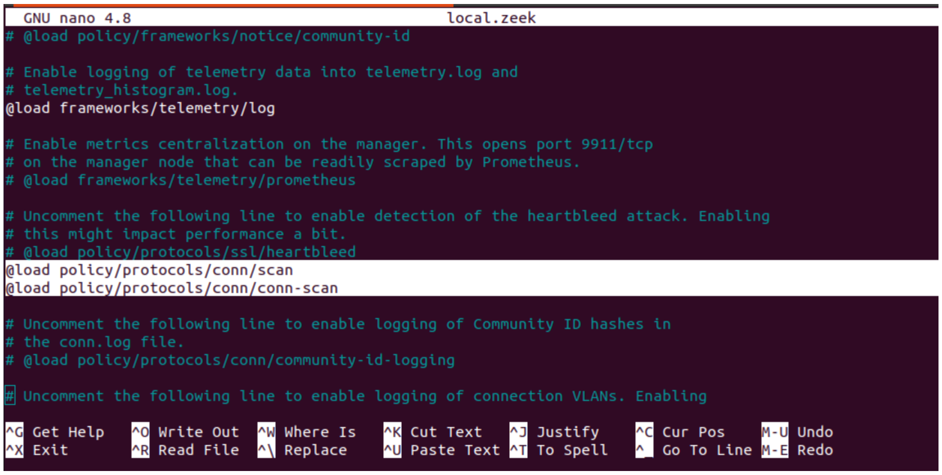


Figure 27: Local.zeek

The figures below show UDP scan.zeek content like:

Source and destination IP addresses.

Scanned ports.

Packet sizes and timings.

Protocol versions.

Any suspicious patterns or anomalies.

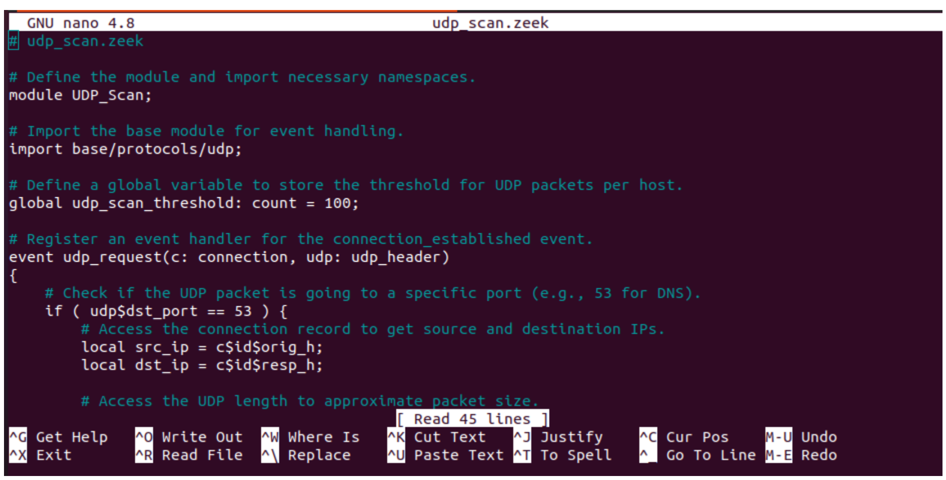


Figure 28: UDP scan.zeek

module UDP scan.

```
  GNU nano 4.8                                          udp_scan.zeek

        # TODO: You may want to perform additional actions based on the UDP scan event.
    }
}

# Log structure for UDP scan events.
type Log::Info UDP_Scan::Info
    = [$src_ip: addr,
       $dst_ip: addr,
       $udp_length: count];

# Define a custom log file for UDP scan events.
export {
    redef Log::ID UDP_Scan::LOG &priority=1;
    redef Log::Columns UDP_Scan::Info &optional;
}

# Register the log writer for the custom log file.
event Log::create_stream(UDP_Scan::LOG: Log::Info);


^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo
```

Figure 29: UDP scan.zeek

custom log file for UDP scan events.

```
  GNU nano 4.8                                          udp_scan.zeek

# Define a global variable to store the threshold for UDP packets per host.
global udp_scan_threshold: count = 100;

# Register an event handler for the connection_established event.
event udp_request(c: connection, udp: udp_header)
{
    # Check if the UDP packet is going to a specific port (e.g., 53 for DNS).
    if ( udp$dst_port == 53 ) {
        # Access the connection record to get source and destination IPs.
        local src_ip = c$id$orig_h;
        local dst_ip = c$id$resp_h;

        # Access the UDP length to approximate packet size.
        local udp_length = udp$length;

        # Log the potential UDP scan event.
        Log::write(UDP_Scan::LOG, [$src_ip=src_ip, $dst_ip=dst_ip, $udp_length=udp_length]);

        # TODO: You may want to perform additional actions based on the UDP scan event.
    }
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo
```

Figure 30: UDP scan.zeek

sudo nmap -sU -p 35 192.168.1.13 192.168.1.180, involves scanning specific IP addresses on network using a UDP scan with root privileges.
nmap: This is a popular network scanning tool used to discover open ports on devices and gather information about the services running on them.
-sU: This option specifies a UDP scan. Unlike SYN scans (TCP), UDP scans are more stealthy but can still be intrusive and raise concerns about malicious intent.
-p 35: This specifies port 35 as the target port to scan.
192.168.1.13 192.168.1.180: These are the IP addresses of the target devices. **sudo**

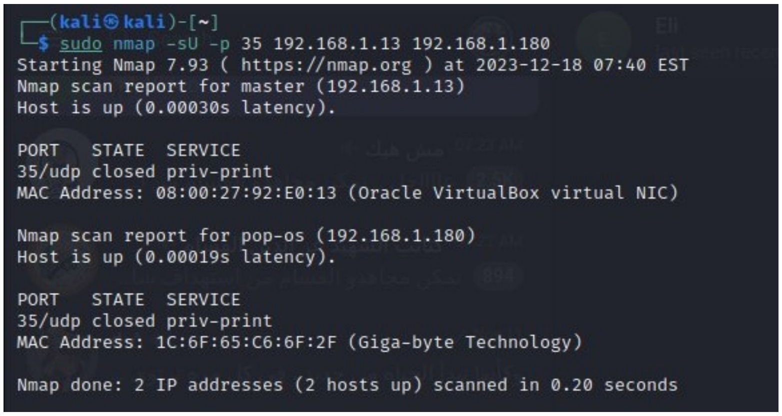


Figure 31: Start UDP scan

nmap -sT -p 35 192.168.1.13 192.168.1.180, involves scanning specific IP addresses on network using a TCP SYN scan with root privileges.
-sT: specifies a TCP SYN scan.

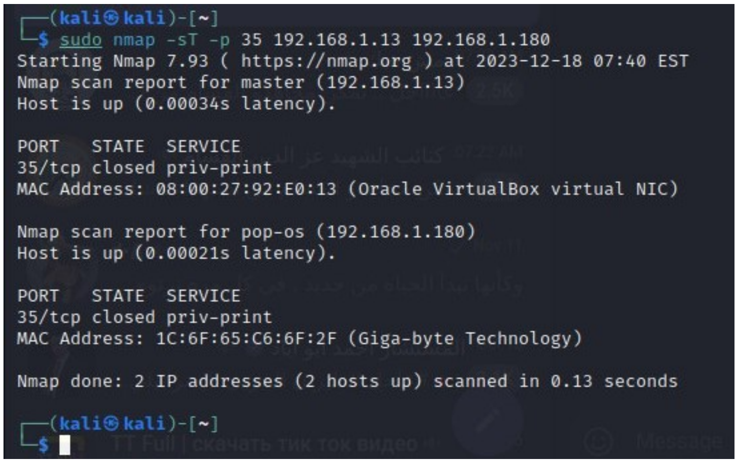


Figure 32: Start TCP scan

In the figures below search for lines containing the **IP"192.168.1.96"** within the log of conn .
conn.log is one of the most important logs Zeek creates.

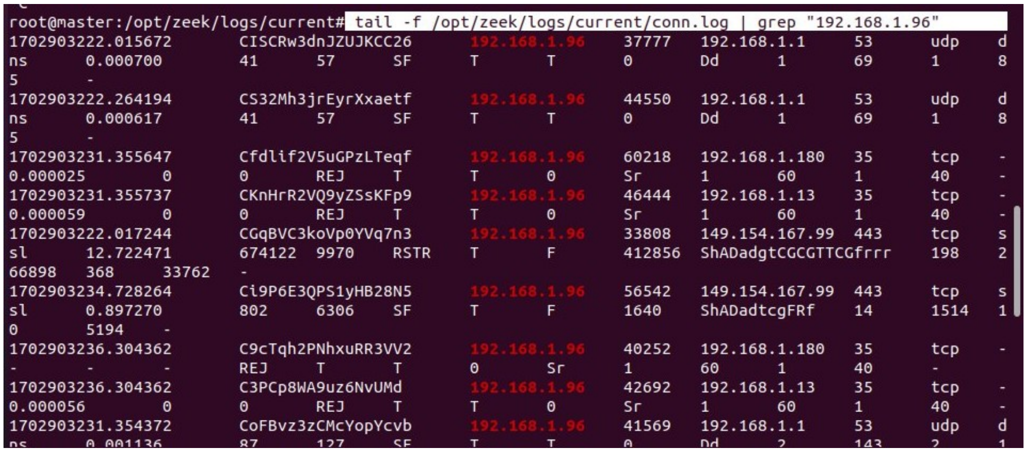


Figure 33: List last lines in conn.log

The figure below show TCP results.

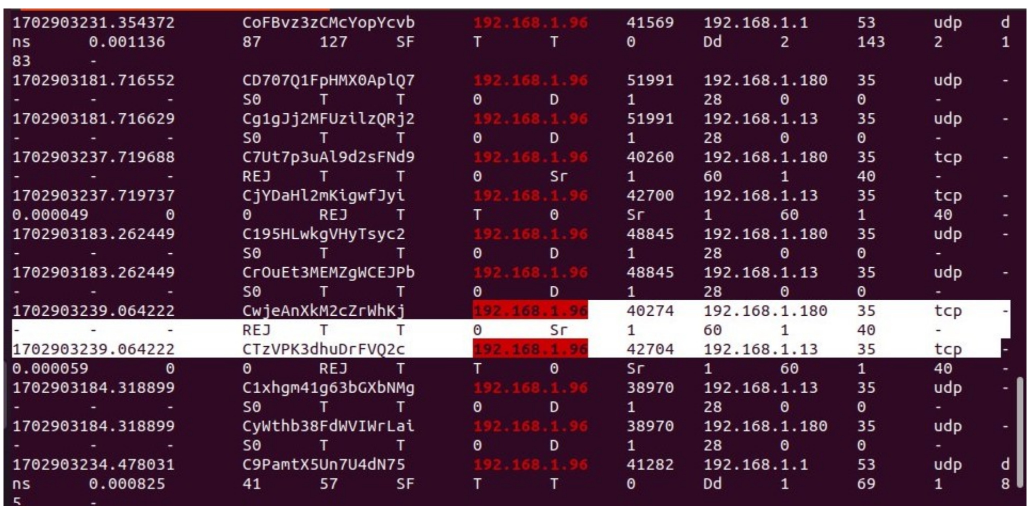
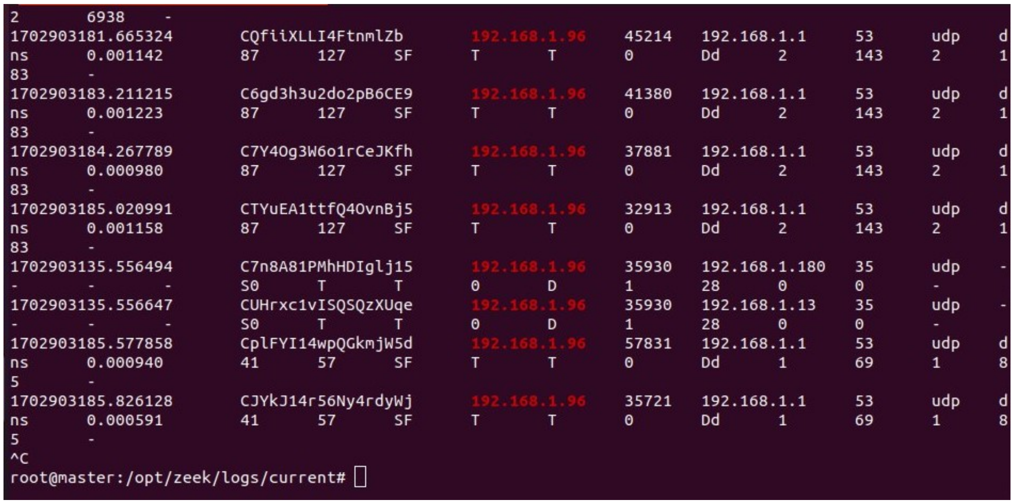


Figure 34: List last lines in conn.log

The figure below show UDP results.



```
2       6938    -
1702903181.665324       CQfiiXLLI4FtnmlZb       192.168.1.96    45214   192.168.1.1     53      udp     d
ns      0.001142        87      127     SF      T       T       0       Dd      2       143     2       1
83      -
1702903183.211215       C6gd3h3u2do2pB6CE9      192.168.1.96    41380   192.168.1.1     53      udp     d
ns      0.001223        87      127     SF      T       T       0       Dd      2       143     2       1
83      -
1702903184.267789       C7Y4Og3W6o1rCeJKfh      192.168.1.96    37881   192.168.1.1     53      udp     d
ns      0.000980        87      127     SF      T       T       0       Dd      2       143     2       1
83      -
1702903185.020991       CTYuEA1ttfQ4OvnBj5      192.168.1.96    32913   192.168.1.1     53      udp     d
ns      0.001158        87      127     SF      T       T       0       Dd      2       143     2       1
83      -
1702903135.556494       C7n8A81PMhHDIglj15      192.168.1.96    35930   192.168.1.180   35      udp     -
-       -       -       S0      T       T       0       D       1       28      0       0       -
1702903135.556647       CUHrxc1vISQSQzXUqe      192.168.1.96    35930   192.168.1.13    35      udp     -
-       -       -       S0      T       T       0       D       1       28      0       0       -
1702903185.577858       CplFYI14wpQGkmjW5d      192.168.1.96    57831   192.168.1.1     53      udp     d
ns      0.000940        41      57      SF      T       T       0       Dd      1       69      1       8
5       -
1702903185.826128       CJYkJ14r56Ny4rdyWj      192.168.1.96    35721   192.168.1.1     53      udp     d
ns      0.000591        41      57      SF      T       T       0       Dd      1       69      1       8
5       -
^C
root@master:/opt/zeek/logs/current#
```

Figure 35: List last lines in conn.log

10 conclusion

This report serves as a comprehensive exploration of Zeek's anomaly-based detection capabilities, showcasing its effectiveness in unraveling the complex tapestry of network traffic. By focusing on protocols such as SSH, HTTP, DNS, and UDP, and honing in on threats like SQL injection and buffer overflow attacks, Zeek emerges as a formidable ally in the ongoing battle against cyber threats. As organizations navigate the digital landscape, Zeek stands as a sentinel, providing insights and defenses to safeguard against the ever-evolving tactics of malicious actors.

References

https://docs.zeek.org/en/master/

https://www.imperva.com/learn/application-security/sql-injection-sqli/

https://www.imperva.com/learn/application-security/buffer-overflow/#what-is-a-buffer-overflow-at

https://www.geeksforgeeks.org/how-to-use-hydra-to-brute-force-ssh-connections/

https://www.httpdebugger.com/http/http_traffic.html

https://www.fortinet.com/resources/cyberglossary/what-is-port-scan