Zeek

Network Security Monitor

Maruthi S. Inukonda

08th March 2024

License

Agenda

- Introduction
- Installation
- Logs
- Scripting

Introduction

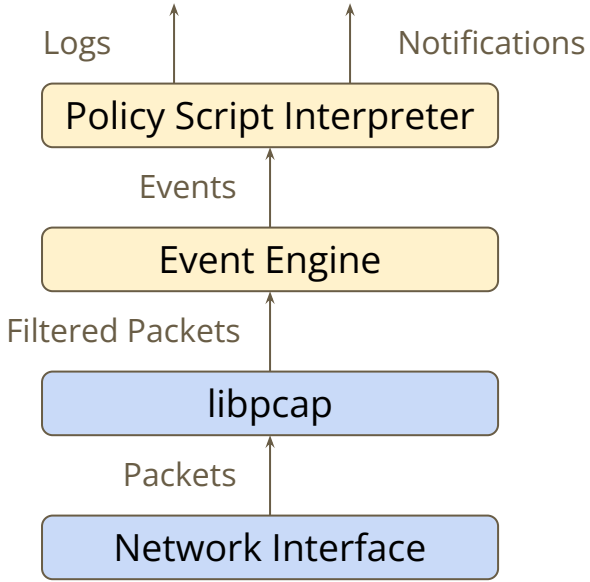
Zeek

- Zeek is a Network Security Monitoring Tool
- Formerly known as Bro
- Developed at Lawrence Berkely National Laborator (LBNL), USA
- Published in Unix Security Symposium 1998

- Code open-sourced under BSD License
- Not an active security device like Firewall, Intrusion Prevention System
- Unobtrusively observes network traffic
- Used for in products for Evidence-Based Network Detection and Response (NDR), and Threat Hunting, Log Management (see Corelight, Security Onion)

Architecture

- Network Interface
  - Tap network link passively, send up a copy of all traffic
- Libpcap
  - Kernel filters down high-volume stream via packet capture library
- Event Engine
  - Distills filtered stream of packets into high-level events
  - Developed in C++
- Policy Script Interpreter
  - Loads scripts written in domain specific policy language (similar to Python)
  - Runs event handlers corresponding to events to take action



Installation

Setup Apt Repository & Install

- Install Long Term Support (LTS) version
  https://docs.zeek.org/en/lts/install.html
- Packages are available for major distros of Linux
- You may also build from source.

```
$ sudo apt-get install -y --no-install-recommends g++ cmake make libpcap-dev

$ echo 'deb http://download.opensuse.org/repositories/security:/zeek/xUbuntu_20.04/
/' | sudo tee /etc/apt/sources.list.d/security:zeek.list

$ curl -fsSL
https://download.opensuse.org/repositories/security:zeek/xUbuntu_20.04/Release.key |
gpg --dearmor | sudo tee /etc/apt/trusted.gpg.d/security_zeek.gpg > /dev/null

$ sudo apt update
$ sudo apt install zeek-lts
```

Select appropriate mail server. If you don't have one, select “No configuration”

Configuration (1/2)

- Zeek runs as root user. Switch to root shell

```
$ sudo su -
#
```

- Configure root's shell environment and reload it

```
# vi /root/.bashrc
.
.
.
export PATH=$PATH:/opt/zeek/bin/

# exec bash
```

- Verify the shell environment

```
# which zeek
/opt/zeek/bin/zeek

# which zeekctl
/opt/zeek/bin/zeekctl

# which zeek-cut
/opt/zeek/bin/zeek-cut
```

Configuration (2/2)

- Find your network interface

```
# ip link show
.
2: enp0s7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode
DEFAULT group default qlen 1000
     link/ether be:ef:fe:ed:be:ef brd ff:ff:ff:ff:ff:ff
3: wlp0s20f3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode
DORMANT group default qlen 1000
     link/ether de:ad:be:ef:de:ad brd ff:ff:ff:ff:ff:ff
```

- Find your network interface

```
# vi /opt/zeek/etc/node.cfg
.
.
[zeek]
type=standalone
host=localhost
interface=af_packet::enp0s7
```

Deploy Zeek

- Deploy Zeek in the background

```
root@terak:~# zeekctl deploy
checking configurations ...
installing ...
removing old policies in
/opt/zeek/spool/installed-scripts-do-not-touch/site ...
removing old policies in
/opt/zeek/spool/installed-scripts-do-not-touch/auto ...
creating policy directories ...
installing site policies ...
generating standalone-layout.zeek ...
generating local-networks.zeek ...
generating zeekctl-config.zeek ...
generating zeekctl-config.sh ...
stopping ...
stopping zeek ...
starting ...
starting zeek ...
```

Verify Zeek

- Check status of zeek running in the background

```
# zeekctl status
waiting for lock (owned by PID 24496) ...
Name      Type       Host          Status    Pid    Started
zeek      standalone localhost     running   25670  04 Mar 22:37:35
```

- Check network statistics

```
# zeekctl netstats
     zeek: 1709573082.768830 recvd=9714926 dropped=0 link=9715276
```

- Check capture statistics

```
# zeekctl capstats
Interface           kpps      mbps      (10s average)
-----------------------------------------
localhost/af_packet::enp0s7 86.2        967.1
```

Logs

Logs

- Currently generated logs are stored in current directory

- Periodically log files are archived in directory named with <date>

```
# ls /opt/zeek/logs/current
capture_loss.log  conn.log  dhcp.log
 dns.log  files.log  http.log  ntp.log ocsp.log
software.log ssl.log  stats.log  stderr.log  stdout.log
telemetry.log weird.log x509.log
```

```
# ls /opt/zeek/logs
2024-03-04  current
```

Logs

- All logs are space/tab separated columns
- For eg., conn.log contains flow records

```
# less /opt/zeek/logs/current/conn.log
.
.
#fields    ts    uid    id.orig_h    id.orig_p    id.resp_h    id.resp_p    proto    service
duration    orig_bytes    resp_bytes    conn_state    local_orig    local_resp
missed_bytes    history    orig_pkts    orig_ip_bytes    resp_pkts    resp_ip_bytes
tunnel_parents
#types    time    string    addr    port    addr    port    enum    string    interval
count    count    string    bool    bool    count    string    count    count    count
count    set[string]
1709602202.400788    CPjdaz38QfyzOsX9c    172.16.0.228    5353    224.0.0.251    5353    udp
dns    2.010773    330    0    S0    T    F    0    D    3    414    0    0-
1709602182.244735    CPR4Mc1NN1j4IBDD72    172.16.0.237    34492    142.250.193.170    443
udp    -    0.428834    3546    4986    SF    T    F    0    Dd    5    3686    9    5238
-
.
.
```

1

2

Difficult to read

Extracting Columns

- zeek-cut is the tool to pick required columns

Using input redirection

```
# zeek-cut -m -d ts uid id.orig_h id.orig_p id.resp_h id.resp_p proto < conn.log
ts    uid    id.orig_h    id.orig_p    id.resp_h    id.resp_p    proto
2024-03-05T07:00:02+0530    CPjdaz38QfyzOsX9c    172.16.0.228    5353    224.0.0.251    5353    udp
2024-03-05T06:59:42+0530    CPR4Mc1NN1j4IBDD72    172.16.0.237    34492    142.250.193.170    443    udp
.
.
```

Using pipe

```
# cat conn.log | zeek-cut -m -d ts id.orig_h id.orig_p id.resp_h id.resp_p proto
ts    uid    id.orig_h    id.orig_p    id.resp_h    id.resp_p    proto
2024-03-05T07:00:02+0530    CPjdaz38QfyzOsX9c    172.16.0.228    5353    224.0.0.251    5353    udp
2024-03-05T06:59:42+0530    CPR4Mc1NN1j4IBDD72    172.16.0.237    34492    142.250.193.170    443    udp
.
```

Scripting

Script init and done (1/4)

- Create a directory to place all our scripts
- zeek_init() event handler is used for initialization
- It is invoked
  - During "`zeekctl deploy`"
  - During "`zeekctl start`"
  - Beginning of "`zeek -r test.pcap`"
- zeek_done() is used for cleanup
- It is invoked
  - During "`zeekctl stop`"
  - Ending of "`zeek -r test.pcap`"

```
# mkdir /opt/zeek/share/zeek/site/myscripts
# cd /opt/zeek/share/zeek/site/myscripts

# vi sample.zeek

event zeek_init() {
     print "Zeek started!";
     # Any other initialization stuff goes here
}

event zeek_done() {
     # Any other cleanup stuff goes here
     print fmt("Zeek stopped!");
}
```

Script init and done (2/4)

- Create a dummy pcap file

```
# tcpdump -w /tmp/test.pcap
```

- Analyze a pcap file

```
# zeek -C -r /tmp/test.pcap sample.zeek
Zeek started!
Zeek stopped!
```

Script init and done (3/4)

- By default no logs are printed during daemon/background mode
- By default the global variable `Log::print_to_log` is set to `Log::REDIRECT_NONE`
- To send all prints to `print.log`, edit our script and assign it with `Log::REDIRECT_ALL`

```
# vi sample.zeek

redef Log::print_to_log = Log::REDIRECT_ALL;

event zeek_init() {
     print "Zeek started!";
     # Any other initialization stuff goes here
}

event zeek_done() {
     # Any other cleanup stuff goes here
     print fmt("Zeek stopped!");
}
```

Script init and done (4/4)

- Add the following line at the **end of the default** site/local.zeek file

```
# vi /opt/zeek/share/zeek/site/local.zeek
.
.
.
@load myscripts/sample
```

- Deploy the script

```
# zeekctl deploy
```

- Check the prints in print.log

```
# less /opt/zeek/logs/current/print.log
```

Variables and Constants

- Global variables are accessible across event handlers and scripts
- Local variables are defined and accessible only inside an event handler
- Constants are accessible in their defined scope (global or local) and cannot be modified

```
# vi sample.zeek

redef Log::print_to_log = Log::REDIRECT_ALL;

global gs = "apple";
const cs = "mango";

event zeek_init() {
     print "Zeek started!";
     local ls = "orange";
     print fmt("gs:%s cs:%s ls:%s", gs, cs, ls);
     gs = "muskmelon";
     ls = "banana";
     print fmt("gs:%s ls:%s", gs, ls);
}

event zeek_done() {
     print fmt("gs:%s cs:%s", gs, cs);
     print fmt("Zeek stopped!");
}
```

Primitive Data Types (1/2)

- String
- Boolean
- Integer (64-bit unsigned)
- Double (64-bit double precision)
- Counter
- Timestamp
- Interval

```
local  ls: string = “pomegranate”;
local  lb: bool = F;
local  li: int = 4;
global ld: double = 9.25;

global gc: count = 0;
local  lt: ts = current_time();
local  linvl: ts = 2.5sec;
```

Primitive Data Types (2/2)

- IP Address
- Subnet
- Port

```
event zeek_init()
{
        .
        .
        .

        local ip : addr = 192.168.1.100;
        local sn : subnet = 192.168.0.0/16;
        local prt : port = 53/udp;
        print(fmt("ip:%s sn:%s port:%d", ip, sn, prt));

}
```

Control Statements (1/2)

- If - else statement

```
local ip: addr = 192.168.1.100;
local sn: subnet = 192.168.0.0/16;
if ( ip/16 == sn )
     print(fmt("ip:%s belongs to sn %s", ip, sn));
```

- for loop
  No need to define loop control variable

```
for (c in "Hello") {
     print fmt("c: %s", c);
}
```

- while loop

```
local i = 0;
while (i <= 5) {
     ++i;
     print "i: ", i;
}
```

Control Statements (2/2)

- `switch case` statement

```
local p : int = 3
switch (i) {
  case 1:
       print "i is 1";
       break;
  case 2:
       print "i is 2";
       break;
  case 3:
       print "i is 3";
       break;
  default:
       print "i is not 1-3";
       break;
  }
```

Composite Data Types (1/3)

- Record types are used to store related information together

```
type MyRecordType: record {
     c: count;
     s: string;
};

event zeek_init()
{
     .
     .
     .
     local lr : MyRecordType = record($c = 9, $s =
"nine");
     print(fmt("lr$c:%d lr$s:%s", lr$c, lr$s));
     .
     .
}
```

Composite Data Types (2/3)

- Vector

```
local vec = vector("one", "two", "three");
print(fmt("vec: %s %s %s", vec[0], vec[1], vec[2]));
for (v in vec){
     print fmt("v: %s", vec[v]);
}
```

- Set

```
local s: set[port] = { 21/tcp, 23/tcp, 80/tcp,
443/tcp };
for (s in s1){
     print fmt("s: %s", s);
}
if (21/tcp in s1) {
     print("port no exists in set");
}
```

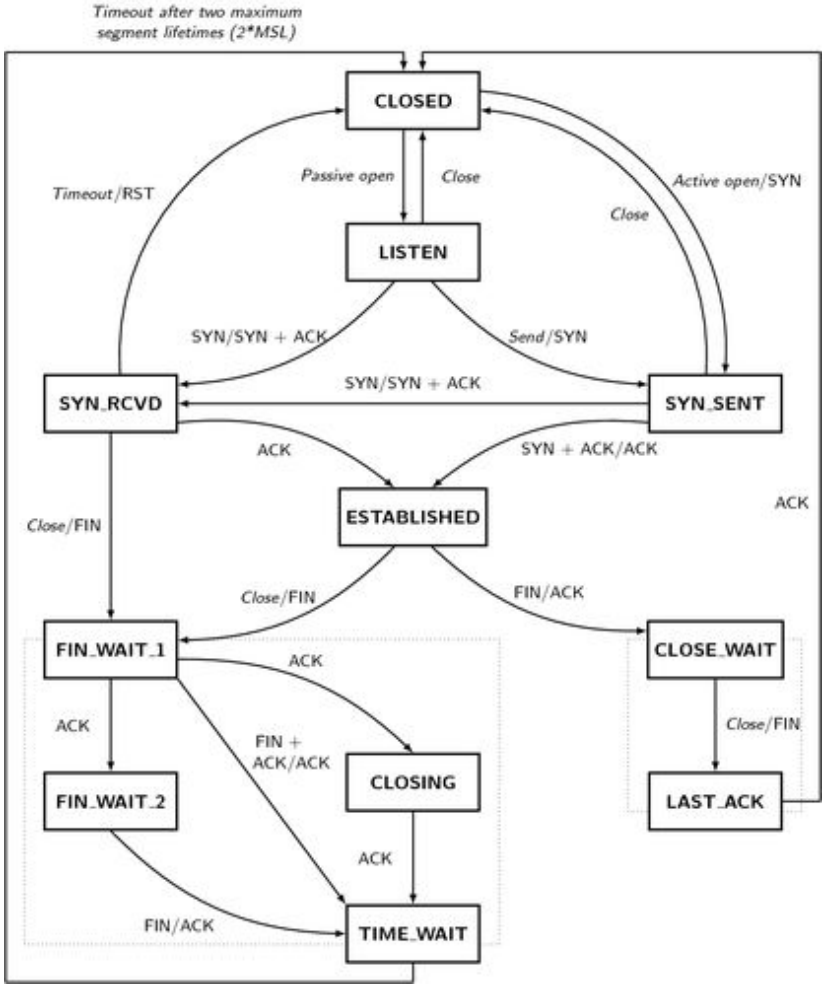
Composite Data Types (3/3)

- Table

```
local t1: table[count] of string = {
     [11] = "eleven",
     [5] = "five",
};
for (t in t1){
          print fmt("t: %s", t);
}
if (11 in t1) {
          print("count exists in table");
}
```

TCP State machine

- `connection_established` : Upon receiving the ACK, both the client and server transition into the **ESTABLISHED** state, signifying a successful connection establishment.
  - Applicable for TCP only
- `connection_finished` : Upon receiving the FIN, both the client and server transition into the **CLOSING** state, signifying a successful connection termination.
  - Applicable for TCP only
- `new_connection` : Upon receiving a packet that makes zeek create a new flow tracking data structure.
  - Applicable for both TCP and UDP



Connection Events

- Add these handlers to the sample.zeek script

```
# vi sample.zeek
.
.
# Applicable for TCP
event connection_established(c: connection)
      {
      print fmt("Zeek connection established from %s:%s to %s:%s",
            c$id$orig_h, c$id$orig_p, c$id$resp_h, c$id$resp_p);
      }

# Applicable for TCP
event connection_finished(c: connection)
      {
      print fmt("Zeek connection finished from %s:%s to %s:%s",
            c$id$orig_h, c$id$orig_p, c$id$resp_h, c$id$resp_p);
      }

# Applicable for TCP and UDP
event new_connection(c: connection)
      {
      print fmt("Zeek new connection from %s:%s to %s:%s",
            c$id$orig_h, c$id$orig_p,  c$id$resp_h, c$id$resp_p);
      }
```

Connection Events

- Run the script on test pcap file

```
# zeek -C -r /tmp/test.pcap sample.zeek
Zeek new connection from 10.10.20.11:34206/tcp to
192.168.137.179:22/tcp

Zeek connection established from 10.10.20.11:56474/tcp to
192.168.137.179:22/tcp

Zeek connection finished from 10.10.20.11:56474/tcp to
192.168.137.179:22/tcp
```

Packet Events

- `tcp_packet` : Generated for every TCP packet.
- `udp_request`: Generated for each packet sent by a UDP flow's originator.
- `udp_reply`: Generated for each packet sent by a UDP flow's responder.
- A very low-level and expensive event due to the volume of traffic.
- Should be avoided in background mode.

```
# vi sample.zeek
.
.
event tcp_packet(c: connection, is_orig: bool, flags: string,
seq: count, ack: count, len: count, payload: string)
      {
      print fmt("Zeek new tcp packet from %s:%s to %s:%s",
            c$id$orig_h, c$id$orig_p,
            c$id$resp_h, c$id$resp_p);
      }

event udp_request(u: connection)
      {
      print fmt("Zeek new udp request from %s:%s to %s:%s",
            u$id$orig_h, u$id$orig_p,
            u$id$resp_h, u$id$resp_p);
      }

event udp_reply(u: connection)
      {
      print fmt("Zeek new udp reply from %s:%s to %s:%s",
            u$id$orig_h, u$id$orig_p,
            u$id$resp_h, u$id$resp_p);
      }
```

Packet Events

- Count the number of TCP/UDP events using zeek
- Count number of TCP/UDP packets using tcpdump utility.

```
# tcpdump -r /tmp/test.pcap udp or tcp | wc -l
reading from file /tmp/test.pcap, link-type EN10MB (Ethernet),
snapshot length 262144
24
```

```
# zeek -C -r /tmp/test.pcap sample.zeek  | wc -l
24
```

Application Protocol-level Events

- Eg. Count the number of http errors
- Redeploy zeek
- Open a non-existent web page
- Check the print.log

```
# vi sample.zeek

global http_404_count: count = 0;
.
.
event zeek_done() {
      print fmt("Zeek stopped. Count: %d", http_404_count);
}

# Applicable for http reply
event http_reply(c: connection, version: string, code: count,
reason: string) {
      if (code == 404)      {
        # Increment the count
        http_404_count += 1;

        # Print information or perform other actions
        print fmt("%d HTTP 404 responses detected.",
http_404_count);
      }
}
```

References

Installation

- Book of Zeek (LTS : v6.0.3)
  https://docs.zeek.org/en/lts/
- Zeek Training Materials/Products
  https://github.com/zeek/zeek-training
- Zeek in Action - Youtube Channel - Many playlists
  https://www.youtube.com/@Zeekurity

Q & A