Suricata, Zeek og DNS Capture exercises

Henrik Lund Kramshoej
hlk@zencurity.com

November 27, 2018

Contents

Preface

This material is prepared for use in *Suricata, Zeek og DNS Capture workshop* and was prepared by Henrik Lund Kramshoej, `http://www.zencurity.com` . It describes the networking setup and applications for trainings and workshops where hands-on exercises are needed.

Further a presentation is used which is available as PDF from kramse@Github
Look for suricatazeek-workshop-exercisesin the repo security-courses.

These exercises are expected to be performed in a training setting with network connected systems. The exercises use a number of tools which can be copied and reused after training. A lot is described about setting up your workstation in the repo

`https://github.com/kramse/kramse-labs`

Prerequisites

This material expect that participants have a working knowledge of TCP/IP from a user perspective. Basic concepts such as web site addresses and email should be known as well as IP-addresses and common protocols like DHCP.

Have fun and learn

Introduction to networking

IP - Internet protocol suite

It is extremely important to have a working knowledge about IP to implement secure and robust infrastructures. Knowing about the alternatives while doing implementation will allow the selection of the best features.

ISO/OSI reference model

A very famous model used for describing networking is the ISO/OSI model of networking which describes layering of network protocols in stacks.

This model divides the problem of communicating into layers which can then solve the problem as smaller individual problems and the solution later combined to provide networking.

Having layering has proven also in real life to be helpful, for instance replacing older hardware technologies with new and more efficient technologies without changing the upper layers.

In the picture the OSI reference model is shown along side with the Internet Protocol suite model which can also be considered to have different layers.

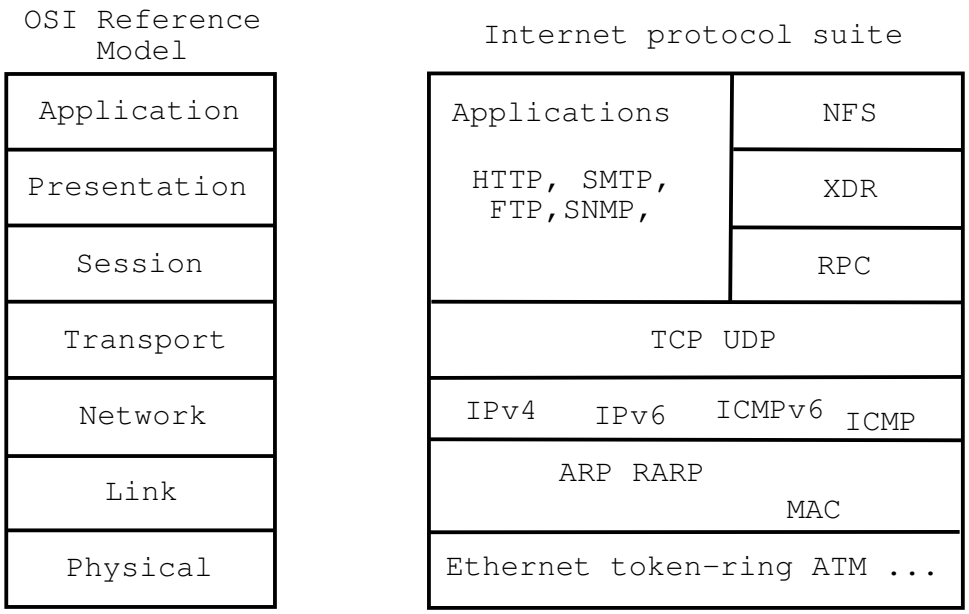


Figure 1: OSI og Internet Protocol suite

Exercise content

Most exercises follow the same procedure and has the following content:

- **Objective:** What is the exercise about, the objective
- **Purpose:** What is to be the expected outcome and goal of doing this exercise
- **Suggested method:** suggest a way to get started
- **Hints:** one or more hints and tips or even description how to do the actual exercises
- **Solution:** one possible solution is specified
- **Discussion:** Further things to note about the exercises, things to remember and discuss

Please note that the method and contents are similar to real life scenarios and does not detail every step of doing the exercises. Entering commands directly from a book only teaches typing, while the exercises are designed to help you become able to learn and actually research solutions.

Exercise 1

Check your VM, run Ansible

Objective:
Make sure your virtual machine is in working order.

We need a Linux server for running the tools.

Purpose:
If your VM is not updated we will run into trouble later.

Suggested method:
Go to `https://github.com/kramse/kramse-labs/tree/master/suricatazeek`

Read the instructions for the setup of a VM.

Hints:
Ansible is great for automating stuff, so by running the playbooks we can get a whole lot of programs installed, files modified - avoiding the Vi editor ☺

Example playbook content

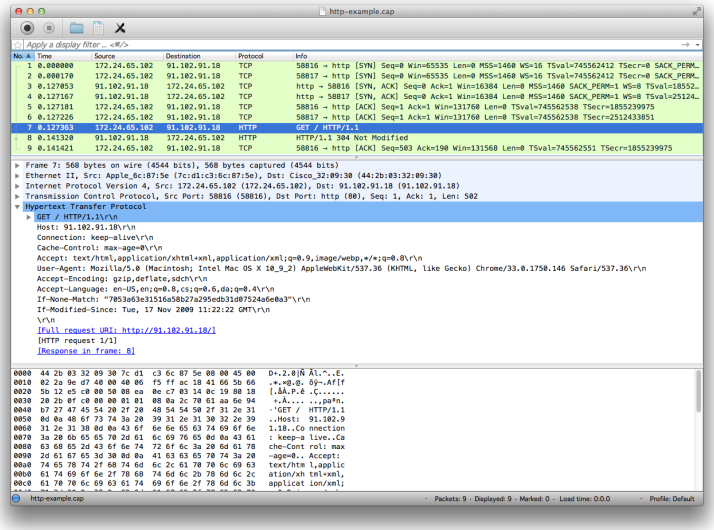
```
apt:
      name: " packages "
    vars:
      packages:
        - nmap
        - curl
        - iperf
        ...
```

Solution:
When you have a updated VM and Ansible running, then we are good.

Discussion:
Linux is free and everywhere. The tools we will run in this course are made for Unix, so they run great on Linux.

Exercise 2

Wireshark and tcpdump



Objective:
Try the program Wireshark locally your workstation, or tcpdump

You can run Wireshark on your host too, if you want.

Purpose:
Installing Wireshark will allow you to analyse packets and protocols

Tcpdump is a feature included in many operating systems and devices to allow packet capture and saving network traffic into files.

Suggested method:
Download and install the program, either download from web server locally or from `http://www.wireshark.org`
Wireshark requires a packet capture library to be installed

Hints:
PCAP is a packet capture library allowing you to read packets from the network. Tcpdump uses libpcap library to read packet from the network cards and save them. Wireshark is a graphical application to allow you to browse through traffic, packets and protocols.

Solution:
When Wireshark is installed sniff some packets. We will be working with both live

traffic and saved packets from files in this course.

If you want to capture packets as a non-root user on Debian, then use the command to add a Wireshark group:

```
sudo dpkg-reconfigure wireshark-common
```

and add your user to this:

```
sudo gpasswd -a $USER wireshark
```

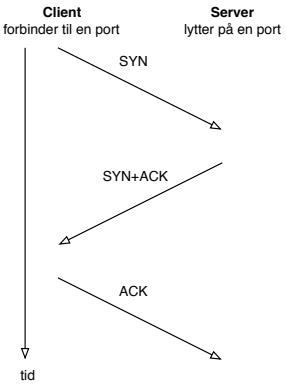
Dont forget to logout/login to pick up this new group.

Discussion:
Wireshark is just an example other packet analyzers exist, some commercial and some open source like Wireshark

We can download a lot of packet traces from around the internet, we might use examples from
`https://www.bro.org/community/traces.html`

Exercise 3

Capturing network packets



Objective:
Sniff packets and dissect them using Wireshark

Purpose:
See real network traffic, also know that a lot of information is available and not encrypted.

Note the three way handshake between hosts running TCP. You can either use a browser or command line tools like cURL

```
curl http://www.zencurity.com
```

Suggested method:
Open Wireshark and start a capture
Then in another window execute the ping program while sniffing

or perform a Telnet connection while capturing data

Hints:
When running on Linux the network cards are usually named eth0 for the first Ethernet and wlan0 for the first Wireless network card. In Windows the names of the network cards are long and if you cannot see which cards to use then try them one by one.

Solution:
When you have collected some packets you are done.

Discussion: Is it ethical to collect packets from an open wireless network?

Also note the TTL values in packets from different operating systems

Exercise 4

Zeek on the web

Objective:
Try Zeek Network Security Monitor - without installing it.

Purpose:
Show a couple of examples of Zeek scripting, the built-in language found in Zeek Network Security Monitor

Suggested method:
Go to `http://try.bro.org/#/?example=hello`

Hints:
The exercise *The Summary Statistics Framework* can be run with a specifc PCAP.

```
192.168.1.201 did 402 total and 2 unique DNS requests in the last 6 hours.
```

Solution:
You should read the example *Raising a Notice*. Getting output for certain events may be interesting to you.

Discussion:
Zeek Network Security Monitor is an old/mature tool, but can still be hard to get started using. I would suggest that you always start out using the packages available in your Ubuntu/Debian package repositories.

They work, and will give a first impression of Zeek. If you later want specific features not configured into the binary packet, then install from source.

Also Zeek uses a broctl program to start/stop the tool, and a few config files which we should look at. From a Debian system they can be found in `/etc/bro` :

```
root@NMS-VM:/etc/bro# ls -la
drwxr-xr-x   3 root root  4096 Oct  8 08:36 .
drwxr-xr-x 138 root root 12288 Oct  8 08:36 ..
-rw-r--r--   1 root root  2606 Oct 30  2015 broctl.cfg
-rw-r--r--   1 root root   225 Oct 30  2015 networks.cfg
-rw-r--r--   1 root root   644 Oct 30  2015 node.cfg
drwxr-xr-x   2 root root  4096 Oct  8 08:35 site
```

Exercise 5

Zeek DNS capturing domain names

Objective:
We will now start using Zeek on our systems.

Purpose:
Try Zeek with example traffic, and see what happens.

Suggested method packet capture file:
Note: a dollar sign is the Linux prompt, showing the command after

```
$ cd
$ wget http://downloads.digitalcorpora.org/corpora/network-packet-dumps/2008-nitroba/nitroba.pcap
$ mkdir $HOME/bro;cd $HOME/bro; bro -r ../nitroba.pcap
... bro reads the packets
~/bro$ ls
conn.log  dns.log  dpd.log  files.log  http.log  packet_filter.log
sip.log  ssl.log  weird.log  x509.log
$ less *
```

Use :n to jump to the next file in less, go through all of them.

Suggested method Live traffic:
Make sure Zeek is configured as a standalone probe and configured for the right interface. Linux used to use eth0 as the first ethernet interface, but now can use others, like ens192 or enx00249b1b2991.

```
root@NMS-VM:/etc/bro# cat node.cfg
# Example BroControl node configuration.
#
# This example has a standalone node ready to go except for possibly changing
# the sniffing interface.

# This is a complete standalone configuration.  Most likely you will
# only need to change the interface.
[bro]
type=standalone
host=localhost
interface=eth0
...
```

Hints:

There are multiple commands for showing the interfaces and IP addresses on Linux. The old way is using `ifconfig -a` newer systems would use `ip a`

Note: if your system has a dedicated interface for capturing, you need to turn it on, make it available. This can be done manually using `ifconfig eth0 up` **Solution:** When you either run Zeek using a packet capture or using live traffic

Running with a capture can be done using a command line such as: `bro -r traffic.pcap`

Using broctl to start it would be like this:

```
// install bro first
kunoichi:~ root# broctl
Hint: Run the broctl "deploy" command to get started.

Welcome to BroControl 1.5
Type "help" for help.

[BroControl] > install
creating policy directories ...
installing site policies ...
generating standalone-layout.bro ...
generating local-networks.bro ...
generating broctl-config.bro ...
generating broctl-config.sh ...
...


// back to Broctl and start it
[BroControl] > start
starting bro
// and then
kunoichi:bro root# cd /var/spool/bro/bro
kunoichi:bro root# tail -f dns.log
```

You should be able to spot entries like this:

```
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       proto   trans_i
     query   qclass  qclass_name     qtype   qtype_name      rcode   rcode_name      AA      TC      RD
1538982372.416180 CD12Dc1SpQm42QW4G3 10.xxx.0.145 57476 10.x.y.141 53 udp 20383 0.045021 www.dr.dk
1 C_INTERNET 1 A 0 NOERROR F F T T 0 www.dr.dk-v1.edgekey.net,e16198.b.akamaiedge.net,2.17.212.93
60.000000,20409.000000,20.000000 F
```

Note: this show ALL the fields captured and dissected by Zeek, there is a nice utility program bro-cut which can select specific fields:

```
root@NMS-VM:/var/spool/bro/bro# cat dns.log | bro-cut -d ts query answers | grep dr.dk
2018-10-08T09:06:12+0200 www.dr.dk www.dr.dk-v1.edgekey.net,e16198.b.akamaiedge.net,2.17.212.93
```

Discussion:
Why is DNS interesting?

Exercise 6

Zeek TLS capturing certificates

Objective:
Run more traffic through Zeek, see the various files.

Purpose:
See that even though HTTPS and TLS traffic is encrypted it often show names and other values from the certificates and servers.

Suggested method:
Run Zeek capturing live traffic, start https towards some sites. A lot of common sites today has shifted to HTTPS/TLS.

Hints:
use broctl start and watch the output directory

```
root@NMS-VM:/var/spool/bro/bro# ls *.log
communication.log  dhcp.log files.log known_services.log packet_filter.log  stats.log
stdout.log x509.log conn.log dns.log known_hosts.log loaded_scripts.log  ssl.log
stderr.log weird.log
```

We already looked at `dns.log`, now check `ssl.log` and `x509.log`

```
root@NMS-VM:/var/spool/bro/bro# grep dr.dk ssl.log
1538983060.546122 CtKYZ625cq3m3jUz9k 10.xxx.0.145 49932 2.17.212.93 443 TLSv12 TLS_ECDHE_RSA_WITH_AES_2
secp256r1 www.dr.dk F -h2 T FzmZCt3o9EYcmNaxIi,FKXcmxQHT3znDDMSj (empty) CN=*.dr.dk,O=DR,L=Copenhagen S
CN=GlobalSign Organization Validation CA - SHA256 - G2,O=GlobalSign nv-sa,C=BE --ok
1538983060.674217 CLjZo51fzuTcvPT0lg 200xxxxb:89b0:5cbf 49933 2a02:26f0:2400:2a1::3f46 443 TLSv12
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 secp256r1 asset.dr.dk F -h2 TFEpW9a1IFe6NTUZNpb,FwV50B4CHIwF1CPlu
(empty) CN=*.dr.dk,O=DR,L=Copenhagen S,ST=Copenhagen,C=DK CN=GlobalSign Organization Validation CA - SH
sa,C=BE --ok
```

Solution:
When you have multiple log files with data from Zeek, and have looked into some of them. You are welcome to ask questions and look into more files.

Discussion:
How can you hide that you are going to HTTPS sites?

Hint: VPN

Exercise 7

Suricata Basic Operation

Objective:
Start using Suricata IDS engine with some traffic.

Purpose:
Show how to get started, meet some obstacles - missing files etc.

Discuss how to solve problems, why do we miss them, how to fix

Suggested method packet capture file:
Note: a dollar sign is the Linux prompt, showing the command after

```
$ cd
$ wget http://downloads.digitalcorpora.org/corpora/network-packet-dumps/2008-nitroba/nitroba.pcap
$ mkdir $HOME/suricata;cd $HOME/suricata;  suricata -r ../nitroba.pcap -c /etc/suricata/suricata.yaml -
l .
... Suricata reads the packets
~/suricata$ ls
eve.json  fast.log  stats.log
$ less *
```

Suggested method live capture:
Make sure the config file `/etc/suricata/suricata.yaml` has the right interface eth0 - or maybe ens192?. Check using `ifconfig -a`

Try starting the service

```
hlk@debian:~$ sudo service suricata start
hlk@debian:~$ cd /var/log/suricata/
hlk@debian:/var/log/suricata$ ls
eve.json  fast.log  stats.log  suricata.log
hlk@debian:/var/log/suricata$

hlk@debian:/var/log/suricata$ tail -3 suricata.log
8/10/2018 -- 17:15:58 - <Warning> - [ERRCODE: SC_ERR_AFP_CREATE(190)] - Can not open iface 'eth0'
8/10/2018 -- 17:15:58 - <Warning> - [ERRCODE: SC_ERR_AFP_CREATE(190)] - Can not open iface 'eth0'
8/10/2018 -- 17:16:19 - <Warning> - [ERRCODE: SC_ERR_AFP_CREATE(190)] - Can not open iface 'eth0'
```

Yeah my network card is called `ens33`, and I should replace eth0 with ens33 in the config file.

```
perl -pi -e "s/eth0/ens33/g" /etc/suricata/suricata.yaml
```

```
hlk@debian:/var/log/suricata$ sudo service suricata stop
hlk@debian:/var/log/suricata$ sudo service suricata start
hlk@debian:/var/log/suricata$ tail -3 suricata.log
8/10/2018 -- 17:23:20 - <Warning> - [ERRCODE: SC_ERR_NO_RULES(42)] - No rule files match the pattern /e
worm.rules
8/10/2018 -- 17:23:20 - <Warning> - [ERRCODE: SC_ERR_NO_RULES(42)] - No rule files match the pattern /e
8/10/2018 -- 17:23:20 - <Notice> - all 2 packet processing threads, 4 management threads initialized, e
```

Hints:
https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Quick_Start_Guide and
https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Basic_Setup

Solution:
When you can start and stop suricata, and it only complains about missing rules, you are done with this exercise.

Discussion:
What was the main problems in this exercise?

Exercise 8

Basic Suricata rule configuration

Objective:
See the Suricata configuration files, and get some rules.

The best IDS is nothing without good rules.

Purpose:
The rules make Suricata useful, and we will learn how to get a ruleset installed, and to keep it updated.

Suggested method:
Check the file `/etc/suricata/suricata-oinkmaster.conf`

It contains this:

```
hlk@debian:/etc/suricata$ cat suricata-oinkmaster.conf
# This is a Debian specific config file for oinkmaster crafted for suricata,
# you should read oinkmaster documentation to modify this file.
# This config is loaded by default from the suricata-oinkmaster-updater binary
# which is called daily from a cronjob by default

skipfile local.rules
skipfile deleted.rules
skipfile snort.conf
use_external_bins = 0

url = https://rules.emergingthreats.net/open/suricata-3.0/emerging.rules.tar.gz
```

Then try running the oinkmaster program in "dry run" with -c

```
root@debian:~# oinkmaster -i -c -C /etc/suricata/suricata-oinkmaster.conf -o /etc/suricata/rules/
Loading /etc/suricata/suricata-oinkmaster.conf
Downloading file from https://rules.emergingthreats.net/open/suricata-3.0/emerging.rules.tar.gz... done
Archive successfully downloaded, unpacking... done.
Setting up rules structures... done.
Processing downloaded rules... disablesid 0, enablesid 0, modifysid 0, localsid 0, total rules 26212
```

If the output looks OK, then re-run without -c and let it update files.

```
root@debian:~# oinkmaster -i -C /etc/suricata/suricata-oinkmaster.conf -o /etc/suricata/rules/
...
```



```
[+] Added files (consider updating your snort.conf to include them if needed):
    -> botcc.portgrouped.rules
    -> botcc.rules
    -> BSD-License.txt
    -> ciarmy.rules
...
    -> emerging-chat.rules
    -> emerging-current_events.rules
    -> emerging-deleted.rules
    -> emerging-dns.rules
    -> emerging-dos.rules
    -> emerging-exploit.rules
    -> emerging-ftp.rules
Do you approve these changes? [Yn]
```

Hints:
You need to restart Suricata for the rules to be found. In the example below I remove the long log with errors, and restart:

```
root@debian:~# service suricata stop
root@debian:~# rm /var/log/suricata/suricata.log
root@debian:~# service suricata start
root@debian:~# cat /var/log/suricata/suricata.log
8/10/2018 -- 17:45:19 - <Notice> - This is Suricata version 3.2.1 RELEASE
```

Solution:
When you have the ruleset downloaded and Suricata is happy when starting you are done with this exercise.

In a real deployment it is advised to automate the update of rules, and also some rules are probably not needed in you environments, YMMV. We will not go through all the rules provided.

Discussion:
Emerging Threats is a well-known ruleset provider, with commercial support.

Whenever there is a new internet wide security incident there are people providing IDS rules in Snort or Suricata format. Since Suricata can read snort rules, this is a good way to add up-to-date rules to your installation.

Note: we haven't mentioned it, but the config files for both Zeek and Suricata allows one to specify your home network.

Checkout the files: Zeek configuration in `/etc/bro/networks.cfg` and Suricata main config `/etc/suricata/suricata.yaml`

```
vars:
  # more specifc is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
    #HOME_NET: "[192.168.0.0/16]"
```


Exercise 9

Configure Mirror Port

Objective:
Mirror ports are a way to copy traffic to Suricata and other devices - for analyzing it. We will go through the steps on a Juniper switch to show how. Most switches which are configurable have this possibility.

Purpose:
We want to capture traffic for multiple systems, so we select an appropriate port and copy the traffic. In our setup, we select the uplink port to the internet/router.

It is also possible to buy passive taps, like a fiber splitter, which then takes part of the signal, and is only observable if you look for signal strength on the physical layer.

Suggested method:
We will configure a mirror port on a Juniper EX2200-C running Junos.

```
root@ex2200-c# show ethernet-switching-options | display set
set ethernet-switching-options analyzer mirror01 input ingress interface ge-0/1/1.0
set ethernet-switching-options analyzer mirror01 input egress interface ge-0/1/1.0
set ethernet-switching-options analyzer mirror01 output interface ge-0/1/0.0
set ethernet-switching-options storm-control interface all
```

Hints:
When checking your own devices this is often called SPAN ports, Mirror ports or similar.

```
https://en.wikipedia.org/wiki/Port_mirroring
```

Cisco has called this Switched Port Analyzer (SPAN) or Remote Switched Port Analyzer (RSPAN), so many will refer to them as SPAN-ports.

Solution:
When we can see the traffic from the network, we have the port configured - and can run any tool we like. Note: specialized capture cards can often be configured to spread the load of incoming packets onto separate CPU cores for performance. Capturing 100G and more can also be done using switches like the example found on the Zeek web site using an Arista switch 7150.

Discussion:
When is it ethical to capture traffic?

Exercise 10

Save Suricata JSON Output in Database

Objective:
Configure a system to read the output files from Suricata EVE logging and save into database system.

This will enable us to use browser based methods and dashboards to analyse more efficiently.

Purpose:
Flat files show that we can collect data, but processing big files when trying to solve problems or handling security incidents is slow. Using databases and document stores like Elasticsearch can help a lot.

Suggested method:
Open the Suricata config file suricata.yml and make sure eve-log is turned on.
`https://suricata.readthedocs.io/en/suricata-4.0.5/output/eve/eve-json-output.html`

```
# Extensible Event Format (nicknamed EVE) event log in JSON format
- eve-log:
    enabled: yes
```

It might be enabled by default. So you can run the (complex) playbook to install database and supporting tools:

```
ansible-playbook -v elasticstack.yml
```

Run the playbook, that installs:

- Logstash for reading the EVE JSON log
- Elasticsearch the database / document store
- Kibana for showing data
- Nginx, because we really should put this in front of Kibana

Hints:
Logstash and Elastic stack are a great way to get started with dashboarding.

However, running a big installation is harder than it looks. Make sure to have multiple servers and good monitoring.

Solution:
When we have a few running installations we are done. Kibana should be available on port 5601 on localhost (127.0.0.1) only though!

Using Firefox visit Kibana on http://127.0.0.1:5601 first time you need to select `logstash-*` as a default index. Note: Kibana is an advanced and powerful tool in itself.

Don't be discouraged if something goes wrong, there are a lot of moving pieces.

A very common problem is the permissions to read files, from logstash log:

```
[2018-10-05T18:22:33,105][WARN ][filewatch.tailmode.handlers.createinitial] failed to open
/var/log/suricata/eve.json: #<Errno::EACCES: Permission denied - /var/log/suricata/eve.json>,
["org/jruby/RubyFile.java:366:in `initialize'", "org/jruby/RubyIO.java:1154:in `open'",
 "/usr/share/logstash/vendor/bundle/jruby/2.3.0/gems/logstash-input-file-4.1.6
 /lib/filewatch/watched_file.rb:204:in `open'"]
```

Discussion:
Making dashboard are an art form. We will NOT start creating beautiful dashboards.

There are a lot of Dashboards available, such as:
https://github.com/StamusNetworks/KTS6

Note: they require Suricata 4.1+ so we cannot use them immediately.

If you want, there is a SELKS LiveCD dedicated to suricata which also includes more tools for administration of rules and getting alerts:
https://www.stamus-networks.com/open-source/

Exercise 11

Suricata Netflow

Objective:
Configure Suricata to do netflow logging

Purpose:
In some cases we dont know what traffic we need to analyze, but if we collect netflow data - summary data about every connection. We can go back and check for specific types of traffic, based on ports, length etc.

Suggested method:

uncomment netflow in the config file `/etc/suricata/suricata.yaml` by removing the "#" in front of this line:

```
#- netflow
```

and restart Suricata.

Hints:
Netflow logging allows efficient logging of summary data, which can be very useful.

Solution:
When you have configured Suricata for netflow, you are done.

Discussion:
Specialized tools exist for collecting and visualizing netflow data. If you have nothing, then Suricata may be a good start.

Exercise 12

Extending Zeek and Suricata

Objective:
Sometimes Zeek and Suricata by themselves will not be enough.

Investigate how to extend Zeek and Suricata, by some examples.

Purpose:
See examples of scripts and rules, evaluate the complexity.

Suggested method:
Get a patch from Henrik for VXLAN support in Suricata. The patch does not need to be installed, but how big is it, how complex is it, could you or your organisation to something similar?

Zeek scripts extend the basic engine, and are a big part of the eco-system. Some 1000s of script lines are already included. Do you have a specific need to analyze in your network which could be implemented in this?

Hints:
Earlier it was quite hard to write C programs for creating and analyzing network traffic. Today we can use the Zeek scripting and Suricata rules to analyze traffic using highly efficient engines.

Solution:
Which tool is easiest to expand, what are you missing from them?

Discussion:

To repeat:
Whenever there is a new internet wide security incident there are people providing IDS rules in Snort or Suricata format. Since Suricata can read snort rules, this is a good way to add up-to-date rules to your installation.

Would you be able to write a rule for something attacking your network?

Exercise 13

Bonus: Indicators of Compromise

Objective:
Indicators of Compromise is a term used for artifacts observed in networks or systems which indicate that a system was compromised.

This could be a known DNS domain where a specifc malware is downloaded from, a specific file name downloaded, a TCP connection to a malware control and command server.

`https://en.wikipedia.org/wiki/Indicator_of_compromise`

Purpose:
The purpose of this exercise is to look at the data gathered and to start planning how one could use this with IOCs to perform after-the-fact analysis of your network.

Goal is to answer how an attack got in, when was the first device compromised etc.

Suggested method:
Look at the data provided by Zeek and Suricata, list the files again.

Which parts will be of greatest interest in your networks? Could some of these facts have helped prevent, restrict, limit or otherwise improve your security stance?

Hints:
I think Suricata and Zeek has excellent value just by turning them on.

Solution:
There is no one solution fits all, results are expected to vary from network to network.

Discussion:
Zeek can include data from other sources, check the intel module
`https://www.bro.org/sphinx/frameworks/intel.html`

and the exercise `https://www.bro.org/current/exercises/intel/index.html`

Would this need to be updated every day to have value? How do we demonstrate return on investment and benefit from looking at traffic?

Exercise 14

Bonus: VXLAN Detection

Objective:
One recent addition to many networks are cloud environments using tunneling and encapsulation to connect islands of containers and virtual systems.

One such protocol named VXLAN can be used without the network people being involved, which can be bad for security. Also it would be easy for an attacker which have compromised a system to use this for exfiltration of data.

So, do you have any VXLAN traffic in your network?

Purpose:
The main idea of this exercise is to talk about unknown traffic, that which you dont even know exist in your network. Some networks have tunnels and IPv6, but the network and security might not be fully aware of this.

Suggested method:
VXLAN traffic will most likely use the default port 4789, which is not used by much other traffic.

VXLAN is also UDP packets, so analysing if a few endpoints use a LOT of UDP might reveal interesting stuff.

Hints:
The conn.log might show you interesting things about such traffic.

Solution:
We dont have a VXLAN tunnel, but it is very easy to add a VXLAN interface to a Linux server, and start sending data out.

Discussion:
Which protocols are the most dangerous, and why?

Appendix A

Host information

- You should note the IP-addresses used for servers and devices
- The web server for installing programs:
  http:// . . . /public/windows/
- Server used for team login: . . .
  Available usernames: team1, team2, ... team10 password: `team`
- You can obtain root access using: `sudo -s`

Available servers and devices:

- IP: . . . - OpenBSD router
- IP: . . . - Your laptop
- IP: . . . - Your laptop VM
- IP: . . . -
- IP: . . . -