

SIEM and Log Analysis exercises

Henrik Kramselund Jereminsen
hlk@zencurity.com

December 10, 2020



Contents

1 Download Debian Administrator's Handbook (DEB) Book 10 min	2
2 Check your Debian VM 10 min	3
3 Investigate /etc 10 min	4
4 Enable UFW firewall - 10 min	6
5 Postman API Client 20 min	8
6 Git tutorials - 15min	9
7 Getting started with the Elastic Stack - 60 min	11
8 Use Ansible to install Elastic Stack	13
9 Making requests to Elasticsearch - 15-75min	15
10 Use a XML library in Python up to 30min	17
11 Clone a Python library StevenBlack/hosts - 30min	19
12 Date Formats 15 min	20
13 Bonus: Grok Debugger 30 min	22
14 Data types – IP addresses 15min	23
15 Data types – IP reputation 15min	24

CONTENTS

16 Bonus: Zeek on the web 10min	25
17 Zeek DNS capturing domain names 15min	26
18 Zeek TLS capturing certificates 15min	28
19 Find Indices in Elasticsearch 15 min	29
20 Zeek Data in Elasticsearch 30 min	33
21 Working with dashboards - xx min	36

Preface

This material is prepared for use in *SIEM and Log Analysis* course and was prepared by Henrik Kramselund Jereminsen, <http://www.zencurity.com> . It describes the networking setup and applications for trainings and courses where hands-on exercises are needed.

Further a presentation is used which is available as PDF from kramse@Github
Look for `siem-log-analysis-exercises` in 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

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

Download Debian Administrator's Handbook (DEB) Book 10 min



Objective:

We need a Linux for running some tools during the course. I have chosen Debian Linux as this is open source, and the developers have released a whole book about running it.

This book is named *The Debian Administrator's Handbook*, - shortened DEB

Purpose:

We need to install Debian Linux in a few moments, so better have the instructions ready.

Suggested method:

Create folders for educational materials. Go to download from the link <https://debian-handbook.info/> Read and follow the instructions for downloading the book.

Solution:

When you have a directory structure for download for this course, and the book DEB in PDF you are done.

Discussion:

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

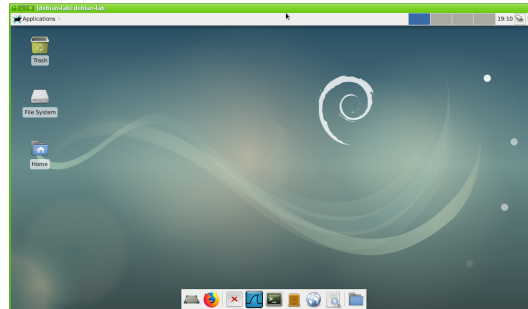
Debian Linux is a free operating system platform.

The book DEB is free, but you can buy/donate to Debian, and I recommend it.

Not curriculum but explains how to use Debian Linux

Exercise 2

Check your Debian VM 10 min



Objective:

Make sure your virtual machine is in working order.

We need a Debian Linux for running tools during the course.

Purpose:

If your VM is not installed and updated we will run into trouble later.

Suggested method:

Go to <https://github.com/kramse/kramse-labs/>

Read the instructions for the setup of a Debian VM.

This is a bonus exercise - only one Debian is needed per team.

Hints:

If you allocate enough memory and disk you won't have problems.

I suggest 50G disk, 2CPU cores and 6Gb memory for this course, if you have this.

Solution:

When you have a updated virtualisation software and a running VM, 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.

Debian Linux allows us to run Ansible and provision a whole SIEM in very few minutes.

Exercise 3

Investigate /etc 10 min

Objective:

We will investigate the /etc directory on Linux. We need a Debian Linux

Purpose:

Start seeing example configuration files, including:

- User database /etc/passwd and /etc/group
- The password database /etc/shadow

Suggested method:

Boot your Linux VMs, log in

Investigate permissions for the user database files passwd and shadow

Hints:

Linux has many tools for viewing files, the most efficient would be less.

```
user@debian:~$ cd /etc
user@debian:/etc$ ls -l shadow passwd
-rw-r--r-- 1 root root  2203 Mar 26 17:27 passwd
-rw-r----- 1 root shadow 1250 Mar 26 17:27 shadow
user@debian:/etc$ ls
... all files and directories shown, investigate more if you like
```

Showing a single file: less /etc/passwd and press q to quit

Showing multiple files: less /etc/* then :n for next and q for quit

Trying reading the shadow file as your regular user:

```
user@debian:/etc$ cat /etc/shadow
cat: /etc/shadow: Permission denied
```

Why is that? Try switching to root, using su or sudo, and redo the command.

Solution:

When you have seen the most basic files you are done.

Also note the difference between running as root and normal user. Usually books and instructions will use a prompt of hash mark # when the root user is assumed and dollar sign \$ when a normal user prompt.

Discussion:

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

Sudo is a tool often used for allowing users to perform certain tasks as the super user. The tool is named from superuser do! <https://en.wikipedia.org/wiki/Sudo>

Exercise 4

Enable UFW firewall - 10 min

Objective:

Turn on a firewall and configure a few simple rules.

Purpose:

See how easy it is to restrict incoming connections to a server.

Suggested method:

Install a utility for firewall configuration.

You could also perform Nmap port scan with the firewall enabled and disabled.

Hints:

Using the ufw package it is very easy to configure the firewall on Linux.

Install and configuration can be done using these commands.

```
root@debian01:~# apt install ufw
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  ufw
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 164 kB of archives.
After this operation, 848 kB of additional disk space will be used.
Get:1 http://mirrors.dotsrc.org/debian stretch/main amd64 ufw all 0.35-4 [164 kB]
Fetched 164 kB in 2s (60.2 kB/s)
...
root@debian01:~# ufw allow 22/tcp
Rules updated
Rules updated (v6)
root@debian01:~# ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
root@debian01:~# ufw status numbered
Status: active
```

To	Action	From
--	-----	----
[1] 22/tcp	ALLOW IN	Anywhere
[2] 22/tcp (v6)	ALLOW IN	Anywhere (v6)

Also allow port 80/tcp and port 443/tcp - and install a web server. Recommend Nginx `apt-get install nginx`

Solution:

When firewall is enabled and you can still connect to Secure Shell (SSH) and web service, you are done.

Discussion:

Further configuration would often require adding source prefixes which are allowed to connect to specific services. If this was a database server the database service should probably not be reachable from all of the Internet.

Web interfaces also exist, but are more suited for a centralized firewall.

Configuration of this firewall can be done using ansible, see the documentation and examples at https://docs.ansible.com/ansible/latest/modules/ufw_module.html

Should you have both a centralized firewall in front of servers, and local firewall on each server? Discuss within your team.

Exercise 5

Postman API Client 20 min

Objective:

Get a program capable of sending REST HTTP calls installed.

Purpose:

Debugging REST is often needed, and some tools like Elasticsearch is both configured and maintained using REST APIs.

Suggested method:

Download the app from <https://www.postman.com/downloads/>

Available for Windows, Mac and Linux.

Hints:

You can run the application without signing in anywhere.

Solution:

When you have performed a REST call from within this tool, you are done.

Example: use the fake site <https://jsonplaceholder.typicode.com/todos/1> and other similar methods from the same (fake) REST API

If you have Elasticsearch installed and running try: <http://127.0.0.1:9200>

Discussion:

Multiple applications and plugins can perform similar functions. This is a standalone app.

Tools like Elasticsearch has plugins allowing decoupling of the API and plugins. Example: <https://www.elastic.co/what-is/elasticsearch-monitoring> and <https://www.elastic.co/what-is/open-x-pack>

Exercise 6

Git tutorials - 15min



Objective:

Try the program Git locally on your workstation

Purpose:

Running Git will allow you to clone repositories from others easily. This is a great way to get new software packages, and share your own.

Git is the name of the tool, and Github is a popular site for hosting git repositories.

Suggested method:

Run the program from your Linux VM. You can also clone from your Windows or Mac OS X computer. Multiple graphical front-end programs exist too.

Most important are Git clone and pull:

```
user@Projects:tt$ git clone https://github.com/kramse/kramse-labs.git
Cloning into 'kramse-labs'...
remote: Enumerating objects: 283, done.
remote: Total 283 (delta 0), reused 0 (delta 0), pack-reused 283
Receiving objects: 100% (283/283), 215.04 KiB | 898.00 KiB/s, done.
Resolving deltas: 100% (145/145), done.

user@Projects:tt$ cd kramse-labs/

user@Projects:kramse-labs$ ls
LICENSE  README.md  core-net-lab  lab-network  suricatazeek  work-station
user@Projects:kramse-labs$ git pull
Already up to date.
```

Hints:

Browse the Git tutorials on <https://git-scm.com/docs/gittutorial> and <https://guides.github.com/activities/hello-world/>

We will not do the whole tutorials within 15 minutes, but get an idea of the command line, and see examples. Refer back to these tutorials when needed or do them at home.

Note: you don't need an account on Github to download/clone repositories, but

having an account allows you to save repositories yourself and is recommended.

Solution:

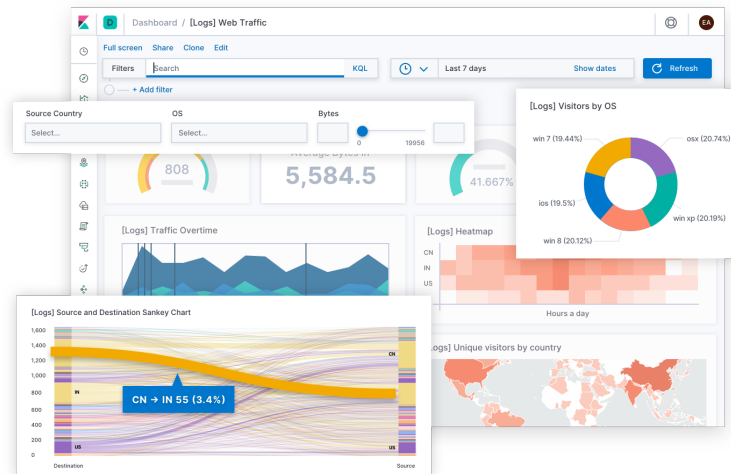
When you have tried the tool and seen the tutorials you are done.

Discussion:

Before Git there has been a range of version control systems, see https://en.wikipedia.org/wiki/Version_control for more details.

Exercise 7

Getting started with the Elastic Stack - 60 min



Objective:

Get a working Elasticsearch, so we can do requests.

Purpose:

We need some tools to demonstrate SIEM systems. Elasticsearch is a search engine and document store used in a lot of different systems, allowing cross application integration.

Elasticsearch uses REST extensively in their application.

Suggested method:

Visit the web page for *Getting started with the Elastic Stack* :

<https://www.elastic.co/guide/en/elastic-stack-get-started/current/get-started-elastic-stack.html>

Read about the tools, and the steps needed for manual installation.

When installing I highly recommend my Ansible based approach - which allows installation in less than 10 minutes automatically. It can also later be re-used in your own organizations to create a proof-of-concept or even production systems.

The ansible is described in exercise 8 on page 13

Hints:

Elasticsearch can store almost anything we like.

The web page for the getting started show multiple sections:

- Elasticsearch - the core engine, this must be done manually or with Ansible
- Kibana - the analytics and visualization platform
- Beats - data shippers, a way to get some data into ES
- Logstash (optional) offers a large selection of plugins to help you parse, enrich, transform, and buffer data from a variety of sources

Each describes a part and are recommended reading.

Solution:

When you have browsed the page you are done.

Discussion:

We could have used a lot of other servers and service, which ones would you prefer?

Exercise 8

Use Ansible to install Elastic Stack

Objective:

Run Elasticsearch

Purpose:

See an example tool used for many projects, Elasticsearch from the Elastic Stack

Suggested method:

We will run Elasticsearch, either using the method from:

<https://www.elastic.co/guide/en/elastic-stack-get-started/current/get-started-elastic-stack.html>

or by the method described below using Ansible - your choice.

Ansible used below is a configuration management tool <https://www.ansible.com/> and you can adjust them for production use!

I try to test my playbooks using both Ubuntu and Debian Linux, but Debian is the main target for this training.

First make sure your system is updated, as root run:

```
apt-get update && apt-get -y upgrade && apt-get -y dist-upgrade
```

You should reboot if the kernel is upgraded :-)

Second make sure your system has ansible and my playbooks: (as root run)

```
apt -y install ansible git python  
git clone https://github.com/kramse/kramse-labs
```

We will run the playbooks locally, while a normal Ansible setup would use SSH to connect to the remote node.

Then it should be easy to run Ansible playbooks, like this: (again as root, most packet sniffing things will need root too later)

```
cd kramse-labs/suricatazeek  
ansible-playbook -v 1-dependencies.yml 2-suricatazeek.yml 3-elasticstack.yml
```

Note: I keep these playbooks flat and simple, but you should investigate Ansible roles for real deployments.

If I update these, it might be necessary to update your copy of the playbooks. Run this while you are in the cloned repository:

```
git pull
```

Note: usually I would recommend running `git clone` as your personal user, and then use `sudo` command to run some commands as root. In a training environment it is OK if you want to run everything as root. Just beware.

Note: these instructions are originally from the course

Go to <https://github.com/kramse/kramse-labs/tree/master/suricatazeek>

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, installing software using APT:

```
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 9

Making requests to Elasticsearch - 15-75min

Objective:

Use APIs for accessing Elasticsearch data, both internal and user data.

Purpose:

Learn how to make requests to an API.

Suggested method:

Go to the list of exposed Elasticsearch REST APIs:

<https://www.elastic.co/guide/en/elasticsearch/reference/current/rest-apis.html>

The Elasticsearch REST APIs are exposed using JSON over HTTP.

Select a category example, Cluster APIs, then select Nodes Info APIs. This will show URLs you can use:

```
# return just process
curl -X GET "localhost:9200/_nodes/process?pretty"
# same as above
curl -X GET "localhost:9200/_nodes/_all/process?pretty"

curl -X GET "localhost:9200/_nodes/plugins?pretty"

# return just jvm and process of only nodeId1 and nodeId2
curl -X GET "localhost:9200/_nodes/nodeId1,nodeId2/jvm,process?pretty"
# same as above
curl -X GET "localhost:9200/_nodes/nodeId1,nodeId2/info/jvm,process?pretty"
# return all the information of only nodeId1 and nodeId2
curl -X GET "localhost:9200/_nodes/nodeId1,nodeId2/_all?pretty"
```

When you can see this works, then feel free to install X-Pack and monitoring plugins

Hints:

Pretty Results can be obtained using the pretty parameter.

When appending `?pretty=true` to any request made, the JSON returned will be pretty formatted (use it for debugging only!). Another option is to set `?format=yaml` which will cause the result to be returned in the (sometimes) more readable yaml format.

Lots of tutorials exist for accessing Elasticsearch

A couple of examples:

- <https://aws.amazon.com/blogs/database/elasticsearch-tutorial-a-quick-start-guide/>
- <https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elastic-stack-on-ubuntu-18-04>

Solution:

When you have seen examples of the API, understand the references with underscore, like `_nodes` and pretty printing you are done.

I recommend playing with Elasticsearch plugins and X-pack.
<https://www.elastic.co/downloads/x-pack>

Note: In versions 6.3 and later, X-Pack is included with the default distributions of Elastic Stack, with all free features enabled by default.

Also Kibana can be used for creating nice dashboards and become applications more or less.

Discussion:

You can also try calling the REST API from Python

Similar to this:

```
#!/usr/bin/env python
import requests
r = requests.get('https://api.github.com/events')
print (r.json());
```

Exercise 10

Use a XML library in Python up to 30min

Objective:

Try using a programming library in the Python programming language.

Purpose:

See how easy it is to produce functionality by re-using existing functions and features available in a popular language.

Suggested method:

Start by getting an XML file. Suggested method is to boot your Linux and run a command like `nmap -p 80,443 -A -oA testfile www.zencurity.com`. Output should be `testfile.xml` and two other files, grepable output `testfile.gnmap` and text output `testfile.nmap`.

Then using Python import a library to parse XML and print a few values from the XML, or all of them.

Recommended values to print from the file:

- Nmap version
- Date of the Nmap run, note either use start and convert from Unix time or startstr which is a string
- Nmaprun args - aka the command line
- Host address
- Ports like from the `<port protocol="tcp" portid="443">`
- Anything you feel like

Hints:

One option is to use the Python ElementTree XML API:

<https://docs.python.org/3/library/xml.etree.elementtree.html>

Also - use Python3!

Solution:

When you can read a file and process it using Python3.

Improvements, you might consider:

- Use Python3 to run the Nmap process

- Create command line parameters for the program, making it more useful
- Pretty print using formatted output

Discussion:

Many examples contain code like this:

Getting child tag's attribute value in a XML using ElementTree

Parse the XML file and get the root tag and then using [0] will give us first child tag. Similarly [1], [2] gives us subsequent child tags. After getting child tag use .attrib[attribute_name] to get value of that attribute.

```
>>> import xml.etree.ElementTree as ET
>>> xmlstr = '<foo><bar key="value">text</bar></foo>'
>>> root = ET.fromstring(xmlstr)
>>> root.tag
'foo'
>>> root[0].tag
'bar'
>>> root[0].attrib['key']
'value'
```

Source:

<https://stackoverflow.com/questions/4573237/how-to-extract-xml-attribute-using-python-elementtree>

What is the point of referring to a specific numbered child, when we specifically have the tags?!

Exercise 11

Clone a Python library StevenBlack/hosts - 30min

Objective:

Find an interesting library on Github, suggest the one called:

<https://github.com/StevenBlack/hosts>

Purpose:

Being able to find libraries on Github can make your life easier.

We will use an example that can download `hosts` files, what is a `hosts` file? **Suggested method:**

Clone the repository in your Linux VM

Browse the `readme.md`

Hints:

Python package manager `pip` makes it easy to install dependencies for a program.

Install the dependencies with:

```
pip3 install --user -r requirements.txt
```

Solution:

When you have read about the tool you are done.

We don't need to run the program today, but feel free to do so.

Discussion:

Are these sources recommendable?

Make sure you read the license, read about the program, possibly inspect the source code.

AND data downloaded also often have restrictions

Exercise 12

Date Formats 15 min

Objective:

See an example of time parsing, and realize how difficult time can be.

Purpose:

System integration often works with different representations of the same data. Time and dates are one aspect we often meet. Realize how complex it is.

Suggested method:

Visit the web pages of an existing tool, Logstash we will use throughout the course and a standard for time and dates.

Write down today's date on a piece of paper, each one does their own.

Then lookup ISO 8601

https://en.wikipedia.org/wiki/ISO_8601

I recommend looking at a specific system, used for processing computer logs: Logstash

<https://www.elastic.co/guide/en/logstash/current/plugins-filters-date.html>

Hints:

When you receive a date there are so many formats, that you need to be very specific how to interpret it.

Parsing dates is a complex task, best left for existing frameworks and functions.

If you decide to parse dates using your own code, then centralize it - so you can update it when you find bugs.

Solution:

When you have a logstash reading a date, or via a Grok debugger on the web

Discussion:

Make sure to visit the web page:

<https://infiniteundo.com/post/25326999628/falsehoods-programmers-believe-about-time>

Did you realize how complex time and computers are?

Then consider this software bug:

"No, you're not crazy. Open Office can't print on Tuesdays."

<https://bugs.launchpad.net/ubuntu/+source/file/+bug/248619>

Linked from

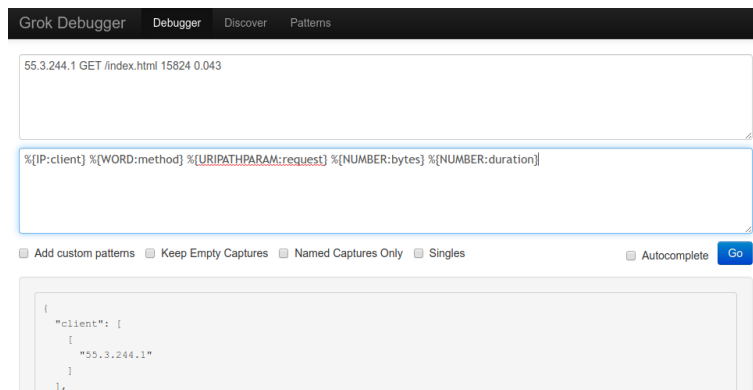
https://www.reddit.com/r/linux/comments/9hdam/no_youre_not_crazy_open_office_cant_print_on/

Because a command file has an error in parsing data, files with PostScript data - print jobs with the text Tue - are interpreted as being Erlang files instead. This breaks the printing, on Tue(sdays).

We will go through this bug in detail together.

Exercise 13

Bonus: Grok Debugger 30 min



Objective:

Try parsing dates using an existing system.

Purpose:

See how existing systems can support advanced parsing, without programming.

Suggested method:

Go to the web application Grok Debugger:

<https://grokdebug.herokuapp.com/>

Try entering data into the input field, and a parsing expression in the Pattern field.

Try the data from <https://www.elastic.co/guide/en/kibana/current/xpack-grokdebugger.html>

Hints:

The expression with greedy data is nice for matching a lot of text:

```
%{GREEDYDATA:message}
```

Try adding some text at the end of the input, and another part of the parsing with this.

Solution:

When you have parsed a line and seen it you are done.

Discussion:

The functionality Grok debugging is included in the tool Kibana from Elastic:

<https://www.elastic.co/guide/en/kibana/current/xpack-grokdebugger.html>

Exercise 14

Data types – IP addresses 15min

Objective:

Find out what IP-addresses really are – just a 32-bit integer for IPv4.

Purpose:

When working with IP addresses, it can be more efficient to use math, than string matching!

Suggested method:

Lets visit the DDS book, page 73 and onwards. Quoted here for ease of use:

Since you know an 8-bit byte can range in value from 0 to 255, you also know the dotted-decimal range is 0.0.0.0 through 255.255.255.255, which is 32 bits. If you count the possible address space, you have a total of 4,294,967,296 possible addresses (the maximum value of a 32-bit integer). This brings up another point of storing and handling IP addresses: **Any IP address can be converted to/from a 32-bit integer value.**

This is important because the integer representation saves both space and time and you can calculate some things a bit easier with that representation than with the dotted-decimal form. If you are writing or using a tool that perceives an IP address only as a character string or as a set of character strings, you are potentially wasting space by trading a 4-byte, 32-bit representation for a 15-byte, 120-bit representation (worst case).

Furthermore, you are also choosing to use less efficient string comparison code versus integer arithmetic and comparison plus bitwise operations to accomplish the same tasks. Although this may have little to no impact in some scenarios, the repercussions grow significant when you're dealing with large volumes of IP addresses (and become worse in the IPv6 world) and repeated operations.

Source: *Data-Driven Security: Analysis, Visualization and Dashboards* Jay Jacobs, Bob Rudis ISBN: 978-1-118-79372-5

Hints:

When working with IP addresses always use libraries

Solution:

When you have pinged the IPv4 address of 2130706433 you are done. `ping 2130706433`

Discussion:

Exercise 15

Data types – IP reputation 15min

Objective:

Find out what IP reputation lists are with some examples.

Purpose:

Identifying bad things can be hard.

We have a concept named, Indicators of Compromise (IoC).

Indicators of Compromise (IOC) any piece of information that can be used to objectively describe a network intrusion, expressed in a platform-independent manner. Say, if a server connects to THIS specific IP, which is KNOWN to be the control and command server for a malware strain, we conclude the device is infected.

For this we can often download files for identification purposes with some reputation.

Suggested method:

We will start with the example of AlienVault's IP Reputation database

Note: links change, and the link in the book does not work!

<http://reputation.alienvault.com/reputation.data>

<https://rdr.io/github/hrbrmstr/netintel/man/Alien.Vault.Reputation.html>

Hints:

Passive DNS systems exist, which allow you to lookup older records, things that have moved.

Maltrail software contains a lot of lists

<https://github.com/stamparm/maltrail>

Solution:

When you have understood that data from others can help your identification efforts, you are done.

Discussion:

We also have used reputation a lot in fighting spam and scam emails.

Exercise 16

Bonus: Zeek on the web 10min

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 specific 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 17

Zeek DNS capturing domain names 15min

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:

Use Nitroba.pcap can be found in various places around the internet

```
$ cd
$ wget http://downloads.digitalcorpora.org/corpora/scenarios/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 ...
...
```

Afterwards you can stop and start as you wish:

```
// back to Broctl and start it
[BroControl] > start
starting bro
[BroControl] > exit
// and then
kunoichi:bro root# cd /var/log/bro/current
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_id	rtt	query	qclass	qclass_name	qtype	qtype_name	rcode	rcode_name
AA	TC	RD	RA	Z	answers	TTLs	rejected	
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 18

Zeek TLS capturing certificates 15min

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

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 19

Find Indices in Elasticsearch 15 min

Objective:

Find the indices in your Elasticsearch

Note: you might not have any.

Purpose:

We need to locate our data. Data in Elasticsearch is saved in **indices**.

Suggested method:

Make sure Elasticsearch is running, with Kibana.

If you have Elasticsearch installed and running try: `http://127.0.0.1:9200`

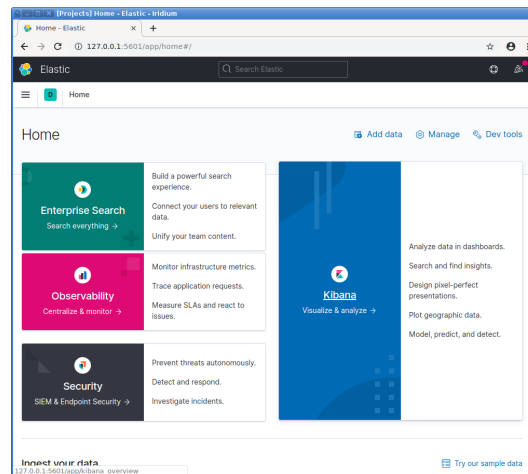
```
$ curl 127.0.0.1:9200
```

should output:

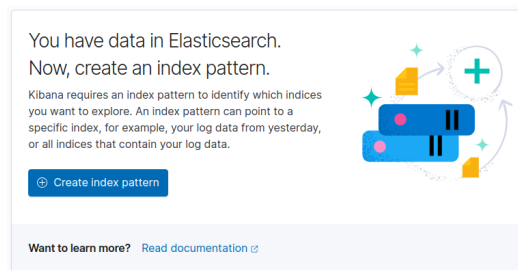
```
{
  "name" : "debian-siem",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "Kyi6e2WuSGq2TcFz0aPnsQ",
  "version" : {
    "number" : "7.10.0",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "51e9d6f22758d0374a0f3f5c6e8f3a7997850f96",
    "build_date" : "2020-11-09T21:30:33.964949Z",
    "build_snapshot" : false,
    "lucene_version" : "8.7.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

If you have Elasticsearch installed and running try: `http://127.0.0.1:5601`

Should show a basic Kibana window.



If this is your first time in kibana, then you will probably see it asking for an index pattern:



This is a short two-step process, and I choose `filebeat-*` as I wanted to play with filebeat data, and already had some available.

Create index pattern

An index pattern can match a single source, for example, `filebeat-4-3-22`, or **multiple** data sources, `filebeat-*`.

[Read documentation](#)

Step 1 of 2: Define an index pattern

Index pattern name

Use an asterisk (*) to match multiple indices. Spaces and the characters `\,/,?,<,>,|` are not allowed.

☐ Include system and hidden indices

✓ Your index pattern matches 2 sources.

filebeat-7.10.0	Alias
filebeat-7.10.0-2020.12.03-000001	Index

Rows per page: 10

Make sure to select a time field:

Create index pattern

An index pattern can match a single source, for example, `filebeat-4-3-22`, or **multiple** data sources, `filebeat-*`.

[Read documentation](#)

Step 2 of 2: Configure settings

Specify settings for your `filebeat-*` index pattern.

Select a primary time field for use with the global time filter.

Time field Refresh

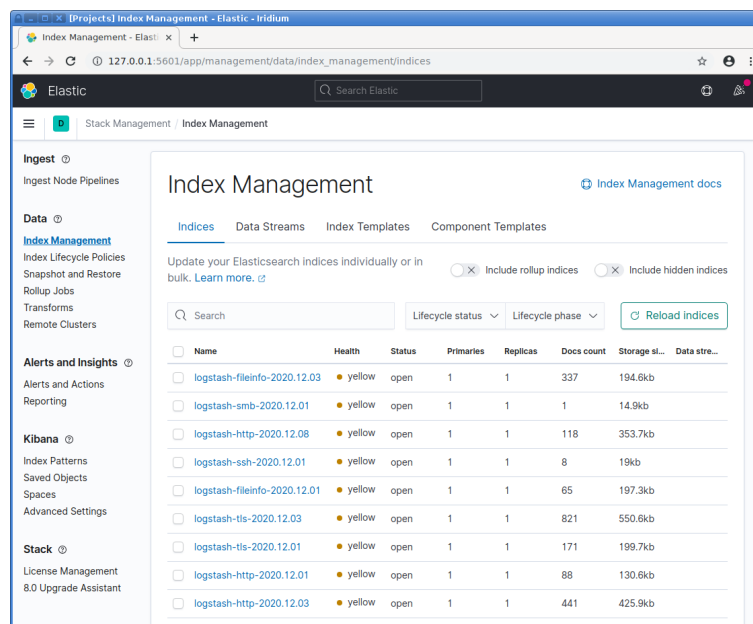
`@timestamp` ▼

[Show advanced settings](#)

[< Back](#) [Create index pattern](#)

When you have done your first index pattern, note the menu on the left, hamburger icon menu.

From there you can choose: Stack Management > Index Management.



Feel free to delete data, or otherwise change data in your own installation.

Hints:

Documentation is available at:

<https://www.elastic.co/guide/en/elasticsearch/reference/7.10/index-mgmt.html>

Solution:

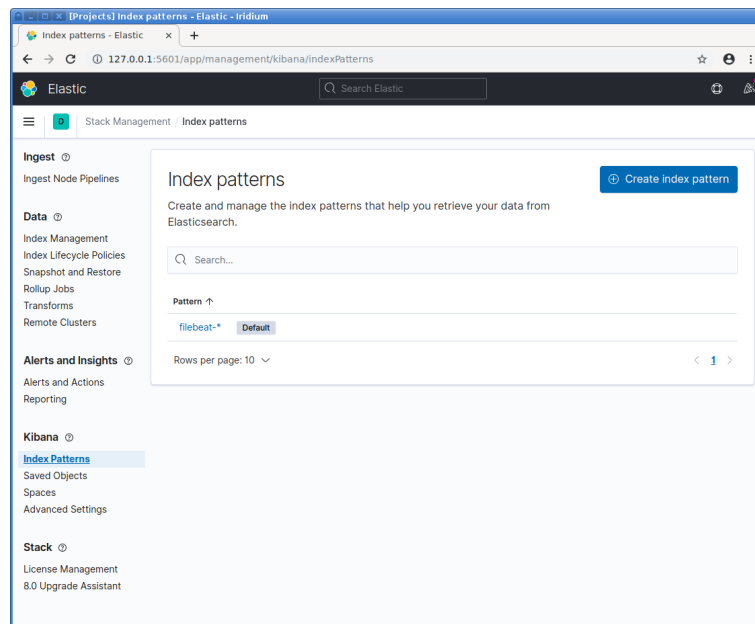
When you have ES and Kibana running, and can access the interfaces, you are done.

Before you leave, make note of the menu: Kibana Index Patterns. These are used for presenting data in Kibana, and you want them to show your data!

Kibana requires an index pattern to access the Elasticsearch data that you want to explore. An index pattern selects the data to use and allows you to define properties of the fields.

Source: <https://www.elastic.co/guide/en/kibana/current/index-patterns.html>

So if you are running filebeat, and data from filebeat, then it is advised to use an index pattern of `filebeat-*`



You can create more of these, and common ones would be:

- `filebeat-*` – all sorts of data ingested by filebeat
- `logstash-*` – all sorts of data ingested by logstash

Discussion:

Elasticsearch is a huge system, watching tutorials and playing with the tools are the recommended way to learn.

Exercise 20

Zeek Data in Elasticsearch 30 min

Objective:

Get data from Zeek files into Elasticsearch.

Purpose:

Having the files with data is great, but to make it more accessible we will load it into ES.

Suggested method:

Git pull the kramse-labs repository and do this automatically.

You can also do the steps manually, following a guide like:

<https://www.elastic.co/blog/collecting-and-analyzing-zeek-data-with-elastic-security>

<https://github.com/kramse/kramse-labs>

If you haven't clone, do this first in your Linux VM:

```
apt -y install ansible git python
git clone https://github.com/kramse/kramse-labs
```

Output should be similar to this:

```
user@Projects:t$ git clone https://github.com/kramse/kramse-labs
Cloning into 'kramse-labs'...
remote: Enumerating objects: 283, done.
remote: Total 283 (delta 0), reused 0 (delta 0), pack-reused 283
Receiving objects: 100% (283/283), 215.04 KiB | 906.00 KiB/s, done.
Resolving deltas: 100% (145/145), done.
user@Projects:t$
```

If you already cloned, get the latest updates:

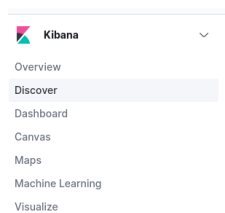
```
user@Projects:kramse-labs$ git pull
Already up to date.
user@Projects:kramse-labs$
```

Run the playbooks for setting up the system as root:

```
cd kramse-labs/suricatazeek
sudo ansible-playbook -v 1-dependencies.yml 2-suricatazeek.yml 3-elasticstack.yml
```

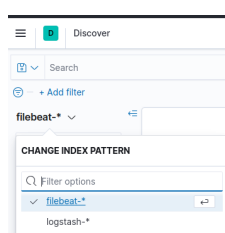
After this, you should have some data in ES/Kibana

Use the discover data menu:

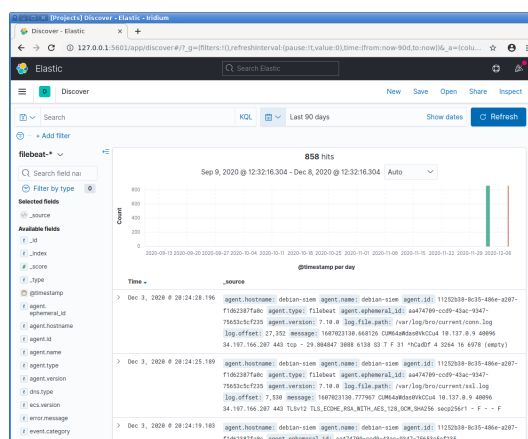


Hints:

Make sure you use the right index pattern. Within discover you can change between them with the menu:



Another hint, for lab systems that are turned off and on, use the time selector at the top and select "Last 90 days". This way you can see older data you imported last time. Production systems continually produce data, but your lab server probably has very little.



Filebeat comes with predefined assets for parsing, indexing, and visualizing your data. To load these assets run the filebeat setup command:

```
filebeat setup -e
```

Solution:

When you have tried the filebeat tool and seen some data you are done.

Discussion:

Exercise 21

Working with dashboards - xx min

Objective:

Get a few dashboard up and running quickly.

Try the dashboard package KTS7 locally your workstation.

Purpose:

There are a lot of Dashboards available, such as:

<https://github.com/StamusNetworks/KTS7>

When learning to create something, a program, document or dashboard – seeing examples help.

Note: these instructions are very dependent on the version of Elasticsearch and Kibana, so check you version first!

```
hkj@debian-siem:~$ dpkg -l | egrep "elasticsearch|kibana"
ii  elasticsearch  7.10.0      amd64      Distributed RESTful search engine built for the cloud
ii  kibana         7.10.0      amd64      Explore and visualize your Elasticsearch data
```

Good news are that the developers of the KTS package has done them for various versions over the years. So look for KTS6 if you are running ES 6.

Suggested method:

Read instructions for installing the dashboards.

<https://github.com/StamusNetworks/KTS7>

Kibana 7 Templates for Suricata Templates/Dashboards for Kibana 7 to use with Suricata. Suricata IDPS/NSM threat hunting and the ELK 7 stack

This repository provides 28 dashboards for the Kibana 7.x and Elasticsearch 7.x for use with Suricata IDS/IPS/NSM - Intrusion Detection, Intrusion Prevention and Network Security Monitoring system

These dashboards are for use with Suricata 6+ and enabled Rust build, Elasticsearch, Logstash, Kibana 7 and comprise of more than 400 visualizations and 24 predefined searches.

The commands are similar to, first clone:


```
git clone https://github.com/StamusNetworks/KTS6.git
cd KTS6
```

then load the dashboards – JSON files:

```
cd API-KIBANA7
curl -X POST "localhost:5601/api/saved_objects/_import" -H 'kbn-xsrf: true' --form file=@index-pattern.ndjson
curl -X POST "localhost:5601/api/saved_objects/_import" -H 'kbn-xsrf: true' --form file=@search.ndjson
curl -X POST "localhost:5601/api/saved_objects/_import" -H 'kbn-xsrf: true' --form file=@visualization.ndjson
curl -X POST "localhost:5601/api/saved_objects/_import" -H 'kbn-xsrf: true' --form file=@dashboard.ndjson
curl -X POST "localhost:5601/api/saved_objects/_import" -H 'kbn-xsrf: true' --form file=@query.ndjson
```

Dont forget to restart Kibana

```
sudo service kibana restart
```

Hints:

Kibana and ES are controlled with web requests, so if you look into the commands, they do HTTP POST requests with JSON files. This allow development of dashboards on one system, and then deploying them easily on others.

Solution:

When you have tried the tool and seen some dashboards you are done.

Note: we might not have any data.

Discussion: