Exercise 09: Intrusion Detection

Jan-Matthias Braun

06. of December, 2022

Having good logging and actually using the information collected are important cornerstones to maintaining a secure deployment. This exercise will just touch on and briefly present exampes of varying complexity as hints towards possible ways to go on.

Introduction

Over the time of the course, one of the essential messages was that you have to be able to watch over what is happening in your network & on your servers.

This essentially calls for monitoring of your system and includes different types of information. Examples are:

- an inventory of your network: which machines are part of it, how is the traffic flow, e.g., scanning with `arpwatch`, using network authentication with 802.1X, etc.;
- an inventory of your services: which services are running, which ports are open, e.g., scanning with `nmap`, `nessus`, or `gvm`, etc.;
- an inventory of traffic in your network, e.g., controlling & logging traffic with firewalls, or (Network) Intrusion Detection Systems.

All of these activities include the collection of specific information, which is worthless, if the collected information is not analysed and not acted upon. Automating the analysis with Intrusion Detection Systems (IDS), and potentially with Intrusion Protection Systems (IPS) for rule based initiation of defence mechanisms, is therefore a corenerstone of good operation.

Today, we will look into exemplary tool to provide a simple understanding of potential uses, issues, and pointers on where to go on.

Technical note: almost all commands in this exercise require administrative privileges. I omitted any sudos and suggest you start an interactive root shell with `sudo -i`, in which you configure the system.

Preparations

Please install the following tools in preparation of today's exercise:

```
apt update
apt install postfix bsd-mailx logcheck sshguard suricata
```

When asked for the role of your postfix mail server, please answer "local only".

Logcheck

The main difficulty with the analysis of the collected information is the overwhelming flood of non-security relevant information, i.e., we need tools to pick up on the important signal. Logcheck is simple tool for sorting the relevance of messages: what can be ignored, what needs to be warned about. It does the by filtering log messages via regular expressions and sending the remaining information out via e-mail. This should ideally happen periodically, e.g., via cron jobs or systemd.

Edit /etc/logcheck/logcheck.conf. Check the REPORTLEVEL setting and change SENDMAILTO to root. Remember that `man pages` can provide you with additional information.

Start your mail service and manually run logcheck to collect information:

```
systemctl start postfix
systemctl enable postfix
systemctl start ssh

sudo -u logcheck logcheck -m root
```

Then check your mail with the `mail` tool:

```
mail
```

Now perform a set of ssh logins, and check again:

```
ssh john@localhost
sudo -u logcheck logcheck -m root
mail
```

- Which information does logcheck provide?

- Why do we need a mail service?
- Is the tool helpful?
- How can you improve the rules?

  Check `man regex` and look into, e.g., `/etc/logcheck/ignore.d.server/ssh`.
- Which dangers do you see with the application of logcheck?

Extended Firewall-Logging

You might think that this should be the exercise's part one, but think about it. :-)

Check the logs continuously with

```
tail -f /var/log/syslog
```

Now perform a standard scan `nmap -sT localhost`

- What can you observe in the logs?
- And what will happen, if you increase the scanner's stealth by using a SYN scan?
- What is the issue here?
- Remark: make sure that postfix is running before trying to answer these questions.

We will mitigate this by extending the coverage of our information by instrumenting the firewall to provide an abundance of logging. This can of course be controlled finely and also, firewalls typically do not see everything happening on the network. What are the limitations here?

Let's increase the log level by issuing the following commands:

```
iptables -I INPUT -j LOG
iptables -I FORWARD -j LOG
iptables -I OUTPUT -j LOG
```

This covers a very wide variety of packages passing through our firewall.

- Now, repeat the two different scan with nmap.
- What do you observe in the logs?
- How can you use this flood of information to get a better view over what is happening in the network?

Note, if you have a strong urge to silence your firewall again, please issue `iptables -F` to flush the existing rules.

- What do you think will a firewall rule like

  ```
  iptables -I INPUT -p tcp \
      -m tcp -m state --state NEW \
      -j LOG --log-level 1 --log-prefix "My log prefix "
  ```

  achieve?

- And which other complex filters might be available for firewall rule creation? Check with `man iptables-extensions` and search the internet.

Service Protection with sshguard

sshguard extends beyond passive log checking to acting on logs. A similar tool is, e.g., fail2ban, but supposedly simpler and more lightweight.

Staying with the example of somebody brute-forcing our ssh server, sshguard can be used to detect and block an ongoing attack, interfacing with the firewall. It therewith forms a very simple working example for an Intrusion Prevention System, blocking a port scan early on.

sshguard plugs between your logs and your firewall. sshguard supports different ways of attaching to your system logs, allowing to be used with many different logging services, understanding more than just ssh messages and furthermore interfacing with many different firewall systems.

Enable the sshguard service: `systemctl start sshguard`

Monitor the sshguard logs with `journalctl -fu sshguard`

For a more comprehensive view of logs, showing ssh as well as sshguard logs, you can also

```
tail -f /var/log/auth.log

# or
journalctl -f -u sshguard -u ssh
```

Now try to connect via ssh with wrong user names and/or password. For this test, you do not need to perform a high-performance brute-force attack with hydra, just failing to log in manually will be sufficient. Important is that you do not connect to localhost/127.0.0.1, as that address is whitelisted and will not provide any relevant change in behaviour.

- Which behaviour is sshguard/the system showing?
- What happens when several attacks are detected?

- Check the configuration in */etc/shguard/sshguard.conf+: Configuration options include whitelists, time outs and thresholds, i.e., how many attacks over which time frame are required to constitue an attack?
- What are the caveats for this kind of protective service with the parameters available?
- How does sshguard block incoming traffic? Use `nft list ruleset` to observe changes to the nftables Linux firewall configuration. Can you read the output?

Suricata

The tools presented until now are rather simple-minded. And while very helpful with small machines, they do not offer the possibility to perfom proper analysis of network traffic.

With suricata, we will look at a system that does exactly that: it uses network capture to find out what is really going on in a network. This goes beyond log coverage at firewall level, that will typically collect information that passes a network boundary on a gateway or router.

Suricata furthermore has the option to be a building block in a larger tool chain, an *Security Information and Event Management*, which goes far beyond a simple intrusion detection system and might use suricata to provide information about security relevant events, together with other sources of information, for example from your firewalls.

Note though that we will not explore a lot of suricata's possibilities, as this would require to either generate lots of network traffic or feed in previously captured network traffic. As some students have announced issue with the available disk space for virtual machines, we will limit ourselves to test cases working on the virtual machine itself, and leave advanced experimentation for self-study.

After installation, we will adjust the configuration, which lives in */etc/suricata*. Open */etc/suricata/suricata.yaml* in your favourite editor and perform the following checks/edits. `$EDITOR /etc/suricata/suricata.yaml` Note, configure your system to set the EDITOR environment variable, or replace $EDITOR manually with gedit, kate, gvim, nano, emacs, or whatever your favourite editor is.

Search for the definition of HOME_NET and set it to just `127.0.0.1/32`, or the proper ip address ranges, as given by `ip a sh`.

Now we need to configure the network interfaces from which to capture packetrs. The relevant sections are ***af-packet*** and **pcap**. Check the local network interfaces with `ip l sh`, you can expect these to be eth0 and maybe eth1, depending on how you configured your virtual machines.

Duplicate the eth0 section in case you have more than one network interface whose traffic you want to have analysed.

We will now perform an update of the rules suricata uses to filter traffic.

```
suricata-update

# You can also use this tool to enable further sources.
# To list the available sources, execute:
suricata-update list-sources

# Now you can add desired sources,
# e.g., suricata-update enable-source malilo/win-malware
# After adding additional sources, you will need
# to re-run suricata-update.

# The update will generate a new suricata.rules file
# in /var/lib/suricata/rules. As the kali linux default
# configuration expects it to be in /etc/suricata/rules,
# just copy it there.
# Or adjust the corresponding default-rule-path variable
# in suricata.yaml
cp /var/lib/suricata/rules/suricata.rules /etc/suricata/rules/
```

Now save & test the configuration file, then restart the suricata service.

```
# First, we will test the configuration
# before actually restarting the service:
suricata -T -c /etc/suricata/suricata.yaml -v

# Then we restart the service:
systemctl restart suricata
```

First test case

The configuration as created will provide logging to, amongst other interfaces, ***/var/log/suricata/fast.log***, which we will monitor during our first test:

```
# Monitor suricata operation
tail -f /var/log/suricata/fast.log

# Test mode run in another terminal
curl http://testmynids.org/uid/index.html
```

- What is testmyndis.org?
- What does this test cover?
- Why would we want a NIDS to react to this test?

A simple rule for detecting pings

If you have a possibility to connect to your kali VM from another machine, then you can use this test do pick up on local network activity.

Create a file */etc/suricata/rules/local.rules* with the following contents:

```
alert icmp any any -> any any (msg:"ICMP ping detected"; sid:1; rev:1;)
```

This rule will alert over icmp packages irrespective of source and destination end points, warning with the specified message.

We now add our own rule file to *suricata.yaml* in the *rule-files* section as

```
rule-files
    - /etc/suricata/rules/local.rules
```

Then we restart suricata with `systemctl restart suricata` and can perform a ping over the captured network interfaces.

Will you be able to pick up on a host-local ping?

How to use suricata's output?

You will have noticed that all alerts result in log file entries, either as plain text entries in *fast.log*, or as json records in *eve.json*. Of course, we could set up logcheck to also investigate our suricata output, but this seems pointless, as suricata is supposed to provide security relavant output.

So where to go next?

Outlook: Security Information and Event Management

Essentially, you want to be able to always paint a picture about what is going on, collecting the information and allowing you to make informed decisions.

Tools like suricata are Network Intrusion Detection Systems (NIDSs), analysing network traffic in life systems or from recorded traffic. For a more complete impression, you will want to include other sources of information and set a system on top of this fused information landscape.

Essentially, the structured *eve.json* is ideally suited for consumption by other systems, for example logstash or jq. Furthermore, suricata can even transmit the logs over the network.

Just as an outlook, I will provide a few pieces of information about how you could go on, e.g.,

- Combining suricata with elastic stack: Digital Ocean article serie The Kibana dashboard
- logstash, again a part of Elastic Stack.
- The [Grafana Dashboard](https://grafana.com/grafana/dashboards/14893-ids-ips/
- EVE: a REST-compliant API implementation