TDDE62 Lab 2: Network Intrusion Detection/Prevention

1 Introduction

There are several reasons why a network intrusion detection system (NIDS) is a very important component in a secure network architecture. Placed at crucial junctions in the network (similar to the firewall), it can see all traffic within the network, as well as outgoing and incoming traffic. A NIDS alerts the network administrator of events in the network that are suspicious and that may require action (using NIDS alert rules). The administrator can then take action, such as blocking incoming IP addresses or shutting down parts of a network. When networks grow it is no longer trivial to control what users of the network are doing. Employees may set up web or ftp servers on their own computers out of convenience, unintentionally opening up for attacks.

When vulnerabilities are detected it may be unfeasible to immediately update all servers within the network, and so it is up to the intrusion detection system to be a quick way of blocking out network traffic that looks like if it is exploiting the vulnerability (using NIDS block rules). We will be using a very well known intrusion detection system called Snort.

If your asking yourself how realistic it is using Snort to block new vulnerabilities, then check the link below (check anyway):

https://ics-cert.us-cert.gov/UPDATE-FBI-Snort-Signatures-Heartbleed-April-2014

The cyber division of the FBI publicly went out and suggested Snort rules that could be used to alert for attacks targeted towards *Heartbleed* as the vulnerability became publicly known in April 2014 (if you download the PDF on the linked site you get the rules).

2 Preparations

We will be using the same virtual machines as in the firewall lab, therefore the first step is to start the virtual machines (check the firewall lab instructions). Once the machines have started log into fw, web.dmz and mail.lan. Resize and place the terminal windows as demonstrated in Figure 1.

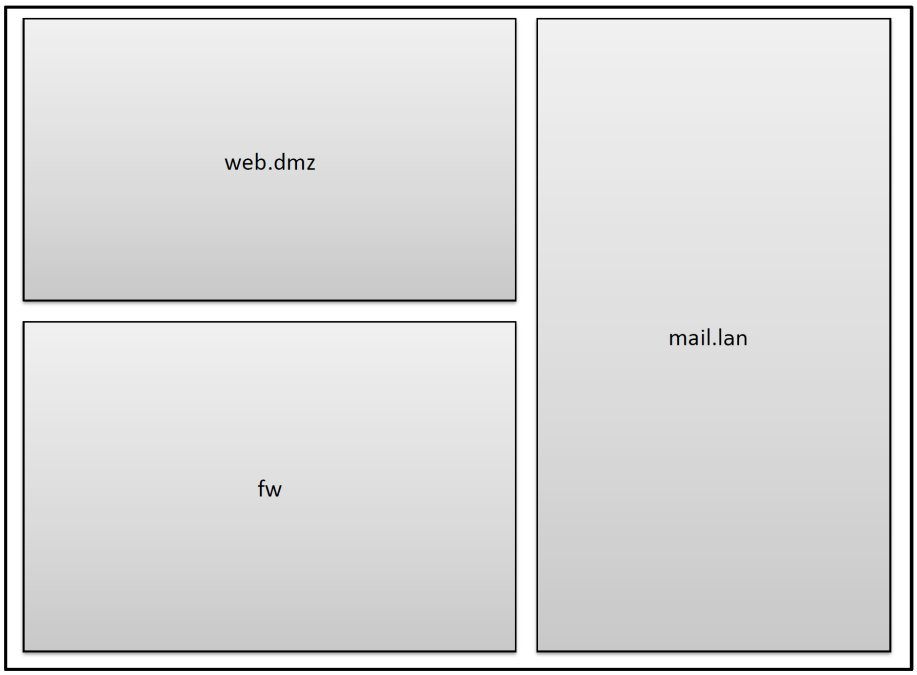

Figure 1: Recommended terminal screen placement.

Complete the following steps:

- **Reset iptables so that the firewall does not block any traffic**.
- On fw: execute the command `systemctl enable snort`. It will take some time to complete.
- On fw: when Snort has been enabled, execute `rm /etc/snort/snort.conf` to remove the default configuration.
- On fw: execute `nano /etc/snort/snort.conf` to open the nano editor and enter the following on a single line at the beginning of the file (it has to be the first line, and it has to be entered correctly, do not copy paste from this document): `output alert_csv: /var/log/snort/alert.csv timestamp,msg,src,srcport,dst,dstport,proto`
- Once the line is in place, save the file by pressing ctrl-o and then hit enter, and then exit nano by ctrl-x.
- On fw: you should now be able to start Snort by executing `snort -i eth1 -c /etc/snort/snort.conf`. If it says "WARNING: No preprocessors configured for policy 0." at the end, and there is a blank line but not a prompt, then you should be fine. If you are given a prompt then something has gone wrong, check that you have followed the instructions above before moving on.

Now Snort is running, and if Snort detects any network traffic that matches your rules, it will write an alert to the file `/var/log/snort/alert.csv`. It is

convenient to monitor this file for changes when debugging your rules, so from the mail.lan terminal use ssh to login to the fw (`ssh 10.19.1.1`). Once logged in, execute `tail -n 100 -f /var/log/snort/alert.csv`. Nothing interesting should happen, as the file is empty and you have no rules in Snort yet.

3 Detection

Essentially we want Snort to alert us if there is network traffic that our security policy does not allow. This could for instance be traffic that relates to known exploits. In order for Snort to be aware of our security policy we write *rules* in `/etc/snort/snort.conf`.

The first rule we will write will alert us if *web.dmz* pings fw. This is not necessarily something that we would care about, but it is a simple rule to check that everything is working correctly. On fw, stop Snort if it is running (press ctrl-c) and enter the following rule on a new line in `/etc/snort/snort.conf`. (Note that it may take up to 10 seconds for Snort to shut down after you press ctrl-c.)

```
alert icmp 10.19.1.10 any -> 10.19.1.1 any (msg:"ICMP Packet";sid:100)
```

The rule says the following: alert if there is an `icmp` packet (could have been `tcp` or `udp` for other rules), coming from the IP address `10.19.1.10` on `any` port (this can be changed to a specific port number, e.g. 22), going to the IP address `10.19.1.1` to `any` port. The part of the rule that is in parentheses are options, in this case the rule only contains the mandatory options, a message (`msg`) to be sent with the alert, and a unique id (`sid`) for this rule (make sure all your rules have a unique `sid` value).

On fw, start Snort again (`snort -i eth1 -c /etc/snort/snort.conf`). If you did not enter the rule correctly you will be given errors, if you have entered it correctly then Snort will start. Now, on *web.dmz* ping the fw by executing `ping 10.19.1.1`. On the terminal that is tailing the alert file (*mail.lan* that has ssh:ed to fw) there should now be a new alert displayed each time a new ping is sent.

This may seem trivial, but you have now set up a NIDS, albeit a very rudimentary one that does not achieve anything useful. You will in Section 4 be given assignments to expand you intrusion detection system to become more useful.

4 Assignment

Create Snort rules so that the following policy is upheld. Your security mechanism should not be overly broad (i.e. alerting all traffic is not accepted). Write a report containing your snort rules and answers to the questions for each task.

4.1 Mandatory

- Task 1: Expand your ping block so that it applies to all source IPs and all destination IPs. Suggest an iptables rule that would block any future

attempts to ping.

- Task 2: A really useful tool for attackers is to do port scanning, in order to see if there is a program listening for traffic on a certain port. There could potentially be vulnerabilities in this program that can be exploited. There are many ways of doing this type of scanning, one way is called FIN scanning. Write rules that detect when any IP is trying to FIN scan your network. How can FIN scanning be blocked in the future?
- Task 3: A common attack is a denial of service (DoS) attack, where the attackers try to flood your servers with a large amount of traffic to slow your servers down, or make them crash. You can read about these quite often in the news. Stage a so called SYN flood attack from *web.dmz* on *mail.lan* by executing `hping3 -S -p 110 --flood --rand-source 10.19.1.141`. Alert that the attack is happening by writing Snort rules, and then discuss what can be done to prevent the attack.
- Task 4: Web servers can easily be set up by anyone on the network. There are many mistakes that can be made in for instance the PHP code of a website that opens up for attackers. The file inclusion attack attempts to read a file from the web server which should not be accessible, for instance executing the following command could be such an attempt: `wget http://10.19.1.140/?file=/etc/passwd`. Alert if somebody is trying to execute this specific attack by writing Snort rules.

4.2 Optional

Google offers remote desktop via Chrome, and the traffic goes over ports that would normally not be associated with remote access. Therefore, if a security policy states that no machine is allowed to be used remotely, it may not be solvable by simply blocking the ports in iptables. The second best option is therefore to raise alerts if this happens, and then find the person using the Chrome remote desktop.

Which Snort rules would be necessary to raise an alert if somebody is using Chrome remote desktop?