Intrusion Detection and Prevention: Network and IDS Configuration and Monitoring using Snort

License

Contents

Preparation

Network monitoring basics

IDS monitoring basics

Writing your own Snort rules

Additional challenges

What to show your tutor for XP rewards

Preparation

Enable VMware player VMs to put the NIC into promiscuous mode... (If you don't know what this means, look up the meaning of promiscuous mode on Google.) From the host OS (the LinuxZ image) run the following in a console (such as Konsole from KDEMenu → System → Terminal → Konsole):

```
sudo chmod a+rw /dev/vmnet*
```

Start Backtrack; if you are using the LinuxZ image use the *Launch VM* desktop icon to start the Backtrack Live Disk virtual machine.

When greeted with the Backtrack "boot:" prompt, hit enter, and again at the boot menu. Start the graphical environment by running:

```
startx
```

Remember: this is a Live DVD VM, so **any changes you make on disk will NOT BE SAVED**. Make sure you manually copy any changes to USB storage.

Take screenshots as you go, to show your tutor, saving a copy to USB storage.

This lab has been designed and tested with openSUSE and Backtrack Linux, but could be completed on most Linux systems with tcpdump, Wireshark, and Snort installed.

Network monitoring basics

This section gives a quick overview of the basics of network monitoring. If you feel you are already familiar with these techniques, keep in mind that these are important foundations, and we will quickly build on these.

On Backtrack, open a terminal console (such as Konsole from KDEMenu → System → Terminal → Konsole).

To view live network traffic, start tcpdump:

```
tcpdump
```

(in each case, once you have seen tcpdump in action displaying packets you can press ctrl-c to exit)

Tip: If running tcpdump generates the error "packet printing is not supported for link type USB_Linux: use -w", then append "-i eth0" to each tcpdump command. (Where eth0 is the name of the interface as reported by ifconfig).

With tcpdump still running, try performing an Nmap port scan against the VM, from the host OS or from another VM (hint: “nmap *IP-address-of-Backtrack-VM*”). Note that tcpdump displays the network activity taking place, including TCP connections and ARP requests.

Tcpdump can format the output in various ways, showing various levels of detail.

Close tcpdump if it is still running (Ctrl-C), and run:

```
tcpdump -q
```

This shows a less verbose version of the output.

While,

```
tcpdump -A
```

Shows the packet content without the information about the source and destination. If you access a web page in a browser (go ahead...), it will display the content, so long as the traffic is not SSL encrypted (for example, if the URL starts with http**s**://).

Note: If you want to access the Web from within Backtrack in the Leeds Met IMS labs, you will need to set Firefox to use the Leeds Met HTTP proxy: which is 192.168.208.51 port 3128. Also note that the fact that all our traffic is going via proxies makes the network traffic a bit more complicated to make sense of, since connections are made to the proxy, rather than directly to the IP address of the web servers. If you try this on another network or access a web server in the lab rather than the Internet, the traffic will be easier to monitor.

Start apache in Backtrack:

```
service apache2 start
```

In Firefox browse to the page by visiting the IP address of the Backtrack VM.

Run:

```
tcpdump -v
```

The above is even more verbose, showing lots of detail about the network traffic.

Now try the port scan again. Note the very detailed output.

It is possible to write tcpdump network traffic to storage, so that it can be analysed later:

```
tcpdump -w /tmp/tcpdump-output
```

While that is running, access a web page from Firefox, then close tcpdump (Ctrl-C).

If you are interested, run “man tcpdump” and read about the many options for output and filtering.

The graphical program Wireshark can also be used to monitor network traffic, and can also read tcpdump output.

Run:

```
wireshark -r /tmp/tcpdump-output
```

Have a look at the recorded network traffic data using Wireshark, and investigate various ways that the data can be displayed. For example, right click one of the HTTP connections, and “follow TCP stream”. Once you have finished, close Wireshark.

We could use tcpdump to do some simple monitoring of the network traffic to detect certain key words.

On the Backtrack VM run:

```
tcpdump -A | grep "www.leedsmet.ac.uk"
```

Tip: if you are using a UK keyboard and Backtrack is configured for US, the “|” symbol is located where “~” is.

Open a web browser, and visit http://leedsmet.ac.uk, note that tcpdump captures *most* network content, and grep can be used to filter it down to lines that are interesting to us.

Take a screenshot as evidence that you have completed this part of the task.

Label it or save it as “IDS-1”.

Note that making sense of this information using tcpdump and/or Wireshark is possible (and is a common sys-admin task), but the output is too noisy to be constantly and effectively monitored by a human to detect security incidents. Therefore we can use an IDS such as Snort to monitor and analyse the network traffic to detect activity that it is configured to alert.

IDS monitoring basics

Start Snort:

```
service snort start
```

Snort should now be running, monitoring network traffic for activity.

Do a port scan of the Backtrack VM (from the host, or another VM).

This should trigger an alert from Snort.

Find the Snort log file, and view the alert. (Hint: the Snort logs are where most log files are stored on a Linux system /var/log/...)

> If there is no alert, you may have to edit /etc/snort/snort.debian.conf, to use the correct interface (for example, eth1 if the output of "ifconfig" does not contain "eth0").

Does the log match what happened? Are there any false positives (alerts that describe things that did not actually happen)?

Try another type of port scan, such as an Xmas Tree scan (*hint: "man nmap"*).

"Follow" the log file by running:

```
tail -f logfile
```

> (where logfile is the Snort alert file)

The tail program will wait for new alerts to be written to the file, and will display them as they are logged.

Note that in Backtrack's default Snort configuration, a "tcpdump" formatted network capture is also stored.

Run:

```
tcpdump -r /var/log/snort/tcpdump.log.*
```

This will print out an overview of all the network activity that has been logged since starting Snort. You can use tcpdump's various flags to change the way it is displayed, or even open the logged network activity in wireshark.

Configuring Snort

Open /etc/snort/snort.conf in an editor; for example:

```
vi /etc/snort/snort.conf
```

> (remember: editing using vi involves pressing "i" to insert/edit text, then *Esc*, ":wq" to write changes and quit)

Scroll through the config file and, take notice of these details:

- In a production environment you would configure Snort to to correctly identify which traffic is considered LAN traffic, and which IP addresses are known to run various servers (this is also configured in snort.debian.conf). In this case, we will leave these settings as is.
- Note the line "var RULE_PATH /etc/snort/rules": this is where the IDS signatures are stored.
- Note the presence of a Back Orifice detector preprocessor "bo". Back Orifice was a Windows Trojan horse that was popular in the 90s.
- We have already seen the "sfportscan" preprocessor in practice, detecting various kinds of port scans.
- The "arpspoof" preprocessor is described as experimental, and is not enabled by default.
- Towards the end of the config file are "include" lines, which specify which of the rule files in RULE_PATH are in effect. As is common, lines beginning with "#" are ignored, which is used to list disabled rule files. There are rule files for detecting known exploits, attacks against services such as DNS and FTP, denial of service (DoS) attacks, and so on.

Add this line (without the quotation marks) below the other include rules:

> "include $RULE_PATH/my.rules"

Save your changes to snort.conf (for example, in vi, press Esc, then type ":wq"). *Remember: if you restart the VM, any changes on disk will not be permanent, so save copies of your work to USB.*

Run this, to create your new rule file:

```
touch /etc/snort/rules/my.rules
```

Edit the file. For example:

```
vi /etc/snort/rules/my.rules
```

Add this line (with your own name), then save your changes:

> alert icmp any any -> any any (msg: "*Your-name*: ICMP Packet found"; sid:1000000; rev:1;)

For example, "alert icmp any any -> any any (msg: "**Cliffe**: ICMP Packet found"; sid:1000000; rev:1;)"

Now that you have new rules, tell Snort to reload its configuration:

```
service snort reload
```

(If after attempting a reload Snort fails to start, you have probably made a configuration mistake, so check the log for details by running: "tail /var/log/syslog")

Due to the new rule you have just applied, sending a simple ICMP Ping (typically used to troubleshoot connectivity) will trigger a Snort alert.

Try it, from the host (LinuxZ):

```
ping Backtrack-VM-IP-Address
```

Check for the Snort alert. You should see that the ping was detected, and our new message was added to the alerts log file.

Take a screenshot as evidence that you have completed this part of the task.

Label it or save it as "IDS-2".

Writing your own Snort rules

Snort is predominantly designed as a signature-based IDS. Snort monitors the network for matches to rules that indicate activity that should trigger an alert. You have now seen Snort detect a few types of activity, and have added a rule to detect ICMP packets. Next you will apply more complicated rules, and create your own.

In addition to the lecture slides, you may find this resource helpful to complete these tasks:

Martin Roesch (n.d.) Chapter 2: Writing Snort Rules - How to Write Snort Rules and Keep Your Sanity. In: *Snort Users Manual*. Available from: <http://biblio.l0t3k.net/ids/en/snort-users-manual/chap2.html> [Accessed 23 January 2012].

In general, rules are defined on one line (although, they can break over lines by using "\"), and take the form of:

header (body)

where header =

action(log,alert) protocol(ip,tcp,udp,icmp,any) src IP src port direction(->,<>)

for example: "alert tcp any any -> any any" to make an alert for all TCP traffic, or "alert tcp any -> 192.168.0.1 23"* to make an alert for connections to telnet on the given IP address

and body =

option; option: "parameter"; ...

The most common options are:

msg: "message to display"

and, to search the packet's content:

content: "some text to search for"

To set the type of alert:

classtype:misc-attack

(where misc-attack is defined in /etc/snort/classification.conf)

To give a unique identifier and revision version number:

sid:1000001; rev:1

So for example the body could be:*

msg: "user login attempt"; content: "user"; classtype:attempted-user; sid:1000001; rev:1;

And bringing all this together a Snort rule could read:

alert tcp any -> 192.168.0.1 110 (msg: "Email login attempt"; content: "user"; classtype:attempted-user; sid:1000001; rev:1;)

This rule looks at packets destined for 192.168.0.1 on the pop3 Email port (23), and sends an alert if the content contains the "user" command (which is used to log on to check email). Note that this rule is imperfect as it is, since it is case sensitive.

There are lots more options that can make rules more precise and efficient. For example, making them case insensitive, or starting to search content after an offset. Feel free to do some reading, to help you to create better IDS rules.

Optional: figure out how the rule could be improved to be case insensitive.

Study the existing rules in /etc/snort/rules and figure out how at least two of them work.

Edit your new rules file:

```
vi /etc/snort/rules/my.rules
```

Add a rule to detect any attempt to connect to a Telnet server. Connections to a Telnet server could be a security issue, since logging into a networked computer using Telnet is known to be insecure because traffic is not encrypted. Make the output message include your name, as we did previously.

Hint: you can combine the information above (tagged with *) to create this rule. Change the IP address to "any", and consider removing any content rules.

Once you have saved your rule and reloaded Snort, test this rule by using Telnet. Rather than starting an actual Telnet server (unless you want to do so), you can simulate this by using Netcat to listen on the Telnet port, then connect with Telnet from the host OS...

On a terminal on the Backtrack VM:

```
netcat -l -p 23
```

Leaving that running, and on a terminal on the host OS (LinuxZ):

```
telnet localhost
```

```
type "hello"
```

Hint: if you have connectivity problems, make sure both systems are on the same subnet (the IP addresses start the same). If you are using VMs, consider setting networking to "bridged".

Look at the alert output, and confirm that your alert has been logged, and it includes your name in the output.

Label it or save it as “IDS-3”.

Create a Snort rule that detects visits to the Leeds Met website, but does not get triggered by general web browsing.

Hints:

Look at some of the existing Snort rules for detecting Web sites, such as those in /etc/snort/rules/community-inappropriate.rules

In the labs, you are likely using the proxy to access the web, so you will need to approach your rules a little differently, you may find you need to change the port you are listening to. Look at the output of tcpdump -A when you access a web page, what does the traffic contain that may point to what is being accessed? Have a look through the output of tcpdump for the text “Host”.

As before, include your name in the alert message.

Save the rule you have created, and take a screenshot of an alert and the rule, as evidence that you have completed this part of the task.

Label it or save it as “IDS-4”.

Problem-based tasks

Enable the arpspoof preprocessor, and get Snort to detect an attempt at arp spoofing.

Take screenshots of configuration changes and an alert, as evidence that you have completed this part of the task.

Label it or save it as “IDS-A1”.

Run Back Orifice server on a Windows VM, and from another Windows VM connect with the client. Get Snort within the Backtrack VM to detect this activity on the Windows VMs. This will involve configuring VMware Player so that the BT VM can put the network card into promiscuous mode (as explained at the beginning of this lab), and making sure that all three VMs are on the same subnet.

Take screenshots of configuration changes, VMs, and an alert, as evidence that you have completed this part of the task.

Label it or save it as “IDS-A2”.

Extra challenging -- setup Snort as an IPS: inline so that it can actually deny traffic, and demonstrate with a rule. You may wish to extend the Leeds Met website rule, so that all attempts to access the website are denied by Snort. After having this work from within the Backtrack VM, see if you can get the Backtrack VM to prevent other VMs running on your computer from accessing the website.

Take screenshots of configuration changes, VMs, an alert, and a denied connection, as evidence that you have completed this part of the task.

Label it or save it as “IDS-A3”.

What to show your tutor for XP rewards

Show your tutor each of the above (in red) evidences. This will be used to allocate XP for the module. Further details of the XP rewards and requirements are available on the *My XP* site.