Computer Science 530 - Lab Assignment #7 - Intrusion Detection -- Fall 2021

Due: Friday November 19, 2021, 4:30 p.m.

Overview

Infrastructure for Lab

You will use two virtual machines, intrusiondectionVM and webserver, which are clones of Snort-on-Centos and f19-heartbleeder respectively. We have included the .ova files again the google drive directory for this lab in case you need it.

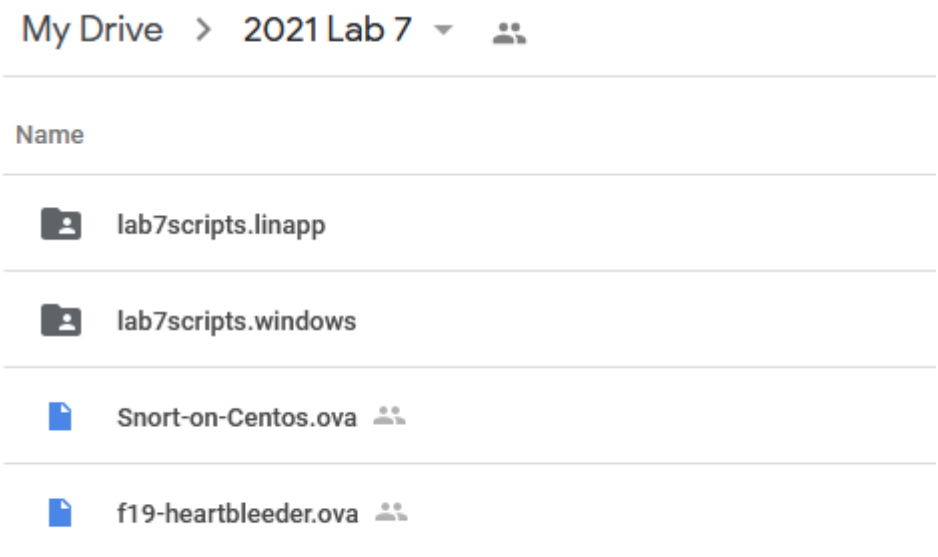
In previous labs, some students had problems setting up their virtual machines because of several issues, including sometimes not running the appropriate scripts to clone machines to configure networks or to establish internal settings for the machines. In other cases, the scripts did not run because vboxmanage was not in the search path for the shell (or command prompt) where the scripts were run. Please review last year's general lab instructions for virtualbox **here,** for guidance on setting up these virtual machines correctly (paying particular attention to the discussion of search path for vboxmanage, as well as the need to create a "base" snapshot for the Snort-on-Centos and f19-heartbleeder virtal machines, before it can be cloned by the populate script. Please be aware that the download instructions for scripts and ova files are different than what is in last years instructions, but the new locations are described in each lab.

Location of files

There are two appliances that you will use in this lab, Snort on Centos, and also the Heartbleeder VM (this second one you have already downloaded for a previous lab). The ova files for both appliances are also available in the CSci530 google drive in the folder for Lab 7 here.

Please note that you may need to login to google drive with your USC account in order to access these files.

(for Heartbleeder, you can use the version previously downloaded if you like, or clone from the pevious installation). The files in the google drive for lab 4 are:



You will note that there is a directoy with scripts (or BATCH files) for this lab. There is a directory for windows machines, and another for Linux and apple systems. Download the scripts from the directory that is relevant to your machine. The scripts in these directories are used to clone the virtual machines (populate), start them (poweron), configure the network between them (construct network, set internal settings for the guest machine (guestOS-internal-setting), power them off, and get rid of them when you are done with the lab (destroy).

You will run these scripts at the appropriate time for the experment nftables below.

Some notes on this instance of fedora Linux

We have already loaded most of the programs you will need for this lab into the virtual appliance. When you start the virtual machine you will be asked to login. The password for both the root and students accounts is "c$l@bLinuX". The third character is the letter "l" as in lab.

Some useful resources for this lab

The following materials may be of use to you in learning about intrusion detection tools that you will use in this lab or elsewhere.

- Slides from last years (2020) lab instruction for the intrusion detection lab
- snort.org
- OSSEC - Open Source Host Based Intrusion Detecton System
- AIDE - Advanced Intrusion Detection Environment
- AIDE - Advanced Intrusion Detection Environment Wikipedia page

Intrusion Detection Lab

CSCI 530 Lab

Intrusion Detection

This lab uses a modification of a virtual machine originally from internetsecurityguru. It's an image of CentOS linux containing a preconfigured copy of the snort intrusion detection system. Ths lab also uses a second machine that runs a web server, for the first to interact with.

Preliminaries

Create, construct network, and power on both machines using provided scripts. The two machines' names are "intrusiondetectionVM" and "webserver". Log in to each as user root and set IP addresses in each as follows.

In intrusiondetectionVM:

```
iptables -F
iptables -X
ifconfig eth0 192.168.1.2
```

In webserver:

```
systemctl  stop  NetworkManager
ifconfig  enp0s3  192.168.1.1
```

On webserver - you may now wish to minimize the window to get it out of the way visually, since the machine will remain passive and you do not need to perform any activities on it. (It serves as a network conversation participant for the benefit of the intrusiondetectionVM machine.)

On intrusiondetectionVM - enter a second virtual terminal and log in there as root a second time. (You can enter a second terminal by keystroke or command. The keystroke is ctrl-alt-F2; the equivalent command is "chvt 2".) Return to the original virtual terminal (ctrl-alt-F1 or "chvt 1"). You can switch your monitor back and forth between them with this way as needed. At any time you can identify in which terminal you are running by executing the "tty" command. We will employ several virtual terminals.

Snort in sniffer mode

Snort can operate as a sniffer. Sniffing is after all an essential prerequisite to intrusion detection-- you must be able to see intrusions in order to be able to detect them! To run snort as a sniffer we want to give it something to sniff. So, on intrusiondetectionVM, let's sniff with snort in virtual terminal 1 while launching a quick ping to webserver from virtual terminal 2. First, returning to virtual terminal 1 (ctrl-alt-F1), start sniffing:

```
cd
snort -v host 192.168.1.2
```

The "host 192.168.1.2" phrase is a filter. It will eliminate confusing, noisy display of busy activity on the network if any, confining it to stuff with the virtual machine as IP source or destination. snort will keep running indefinitely. Now switch to virtual terminal 2 and ping:

```
ping -c 1 -s 4  -p "41424344" 192.168.1.1
```

This says send a single ping (icmp) message containing 4-bytes of payload consisting of ABCD ("41424344" are their ascii codes in hex), for easy visual identifiability in snort. Go back to snort in virtual terminal 1. Terminate it by pressing ctrl-C. (Be patient, I found it to take an inexplicably long time when duplicating these instructions. Or be impatient, ctrl-Z puts snort in the background then "killall -9 snort" termintates it.) Look at what snort captured. At the end snort prints some packet statistics which may scroll the packets off the screen. If so, press shift-PageUp to scroll backward in the screen buffer and view the packets.

There are two other snort command options of interest, -d and -e. From the man page:

-v Be verbose. Prints packets out to the console.

-e Display/log the link layer packet headers.

-d Dump the application layer data when displaying packets in verbose...mode.

So repeat the investigation using -e and -d as follows:

```
snort -ev host 192.168.1.2
```

```
snort -dev host 192.168.1.2
```

in succession, re-pinging from virtual terminal 2 each time (use up arrow to recall the ping command instead of retyping it). snort with -v, -ev, and -dev gives as output different combinations of ethernet frame header, IP packet header, icmp message header, and icmp message data. (Review the "SANS Institute "TCP/IP and tcpdump Pocket Reference Guide" to make sure you know what these are and can identify them in snort's output when you see them).

Snort in logger mode

Snort can save and later re-read what it captures, much as tcpdump does. In fact, snort saves in the same file format. Snort, tcpdump, wireshark, and a number of other programs can thus all share and cross read each other's files. By default snort generates its own names for capture files, you don't have to name them. But it wants to put them in a directory and if you want other than the default (/var/log/snort/) you must create the receiving directory and identify it to snort. Create it:

```
mkdir ./log
```

(In this exercise we make our own log file. More generally snort uses /var/log/snort/ by default.) Then log some stuff:

```
snort -dev -l ./log
```

Wait a while to let traffic accumulate then interrupt with ctrl-C. (There may be no traffic, so if you want to generate some, from the other virtual terminal you can browse a website using the character mode browser lynx, e.g., "lynx 192.168.1.1". After the page has loaded, quit lynx by pressing q then y.) Check that snort deposited a capture file in the receiving directory:

```
ls -l ./log
```

Its name is snort.log.*tttttt* where *tttttt* represents the time of capture. Use the "file" command to find out what kind of content it has:

```
file ./log/snort.log.tttttt
```

It's a tcpdump capture file. Just to make sure:

```
tcpdump -nn -r ./log/snort.log.tttttt
```

Yes, tcpdump can read it alright. And snort too can read/play it back:

```
snort -r log/snort.log.tttttt | less
```

Scroll up and down, take a look around, then press q to exit less.

Snort in ids (intrusion detction) mode

When merely sniffing and logging, snort is passive. It doesn't do anything about it. But it is capable of reacting, if only you define what to react to and how to react. That's what rules do. You convey rules to snort by putting them in files and pointing snort to the files. Run snort now, in virtual terminal 1, pointing it to configuration file snort.conf which in turn tells it to pay attention to the rules in a series of about 40 rules files found in /etc/snort/rules:

```
snort -dev -l ./log -L bigping -h 192.168.1.0/24 -c /etc/snort/snort.conf host 192.168.1.2
```

This also takes control of the name of the logfile, specifying "bigping". Again lauch a ping from virtual terminal 2 but, using ping's -s option, make the ping packet abnormally huge:

```
ping -c 1 -p "41424344" -s 4000 192.168.1.1
```

Now, after terminating snort back in virtual termina 1, examine results in the log directory. First, of course, the large ping should have been logged. Replay it:

```
snort -r ./log/bigping.tttttt | less
```

There's the big fat echo request, bloated with ABCDs, and its big fat echo reply. More interesting, note there's a file named "alert" in the log directory. Check what's at the bottom of that file:

```
tail ./log/alert
```

It contains something like:

```
[**] [1:499:4] ICMP Large ICMP Packet [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
10/16-21:02:19.445399 0:3:25:28:52:C4 -> 0:C:29:1B:AE:7B type:0x800 len:0xFCA
192.168.1.140 -> 192.168.1.114 ICMP TTL:128 TOS:0x0 ID:58836 IpLen:20 DgmLen:4028
Type:0 Code:0 ID:16 Seq:0 ECHO REPLY
[Xref => http://www.whitehats.com/info/IDS246]
```

which was written in response to seeing the huge ping. This alert's presence in the file is in reaction to the ping. There's no reaction to a regular ping though; the ping has to be big in order to get a rise out of snort. How big? Who says? This must be the product of a rule somewhere that says so. We said above that we think the rules come from files in /etc/snort/rules. Here's an attempt to find the rule that operated above:

```
grep "Large ICMP" /etc/snort/rules/*
```

Here, grep is searching for a fragment of the text seen in our alert message, embedded somewhere among the rules files. grep's output is like this:

```
/etc/snort/rules/icmp.rules:alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Large
ICMP Packet"; dsize:>800; reference:arachnids,246; classtype:bad-unknown; sid:499; rev:4;)
/etc/snort/rules/sid-msg.map:499 || ICMP Large ICMP Packet || arachnids,246
```

Looks like there's a relevant rule in file icmp.rules. What threshold size defines what's alertable and what's not? Test your answer by firing pings, while snort is running, at your hypothetical threshold size and one more or one less. Each time look in the ./log/alert file afterward to see if there's a resulting alert there or not. Find the ping "-s <size>" option value that is the boundary condition for alerting.

Now let's do something more normally considered intrusive behavior, a port scan. In virtual terminal 1:

```
snort -dev -l ./log -h 192.168.1.0/24 -c /etc/snort/snort.conf host 192.168.1.2
```

And in virtual terminal 2, here's the port scan:

```
nmap -v -sT 192.168.1.1
```

When it's done, look for any entries just added to ./log/alert provoked by our port scanning.

Now let's write a customized rule of our own. We've been slinging a lot of ping packets containing "ABCD." How about a rule that will raise an alert about them for that reason (not because they be huge or tiny, just because of ABCD)? We must write our own rule and put it in the "my customized rules" file. That file is /etc/snort/rules/local.rules. To that file, append the following:

```
alert icmp any any -> any any (msg:"ABCD embedded"; content:"ABCD";)
```

then restart snort (so that it will re-read its config files and implement the new rule):

```
service snort restart
```

So in virtual terminal 1 start snorting:

```
snort -dev -l ./log -L abcd -h 192.168.1.0/24 -c /etc/snort/snort.conf host 192.168.1.2
```

and in virtual terminal 2 start pinging:

```
ping -c 1 -p "41424344" 192.168.1.1
```

Find the alerts at the bottom of ./log/alert that indicate "ABCD embedded" for both the ping (echo) request and the ping reply.

Translating a snort textfile "alert" into a swatch email alert

Alerts are supposed to get attention. One that just inserts text into a file silently may seem no alert at all. Can't we email the administrator when a port scan occurs, for instance? Well no, snort doesn't do email, but yes, other programs can. swatch (simple watchdog) is such a program. It can dynamically watch any file and take arbitrary action whenever some preconfigured text appears in it. Sending some email could be that resulting action. While swatch won't watch for port scans and snort won't email, swatch will email when a "port scan occurred" message appears in a file and snort can provide that message whenever there's a port scan. Voila. They are complementary.

Let's send the administrator (root) an email whenever the above ping-provoked event occurs (namely, "ABCD embedded" shows up in ./log/alert). Let's use 4 virtual terminals:

virtual terminal 1 - for running snort
virtual terminal 2 - for running swatch
virtual terminal 3 - for executing ping

(By the way, when working with lots of virtual terminals you could get confused which one you're working in. The "tty" command will tell you.)

In virtual terminal 1 get snort running:

```
snort -dev -l ./log -L alpha -h 192.168.1.0/24 -c /etc/snort/snort.conf host 192.168.1.2
```

In virtual terminal 2, configure and get swatch running. To configure, create a file in your home directory (/root) named swatchconfig with these contents:

```
watchfor /ABCD embedded/
   exec /bin/echo "ABCD appeared" | /bin/mail -s "ABCD again!" root
```

then run swatch as follows:

```
swatch -c ~/swatchconfig -t /root/log/alert
```

This says, "Continuously observe the content of /root/log/alert. Per instructions in ~/swatchconfig, perform what it tells me to do whenever I see what it tells me to watch for." In this case, ~/swatchconfig tells swatch to watch for the magic phrase "ABCD embedded" and to send off an email message in response.

In virtual terminal 3, log in and pull the trigger by running ping as before.

```
ping -c 1 -p "41424344" 192.168.1.1
```

Now, as you're running as root, check the administrator's (your) mail:

```
mail
```

"mail" is the ols command line tool for sending, and in this case reading, a user's mail. It is the historical antecedent to later email systems.

"ABCD" isn't very meaningful but you could use the technique for more meaningful and focused targets. A snort article from RedHat Magazine points out, "Close analysis of the protocol in use can turn up signature events. For example, a user logging into an ftp server may pass the string "`user root`". A rule can be written to look for that specific string on FTP's port."

Clean up - if you wish to revert back, please remove the swatchconfig file from your home directory, and use an editor to delete your custom rule about ABCD from /etc/snort/rules/local.rules.

After you have performed the above lab components, answer the following questions.

1. What was the result of your test to determine the ping threshold size in the "Snort in ids mode" section above? That is, what's the smallest value for ping's "-s <size>" that triggers an alert?

2. In the /var/log/snort directory I find one file named alert and several files whose names begin with snort.log. What is the difference between their contents and purposes?

3. Here is a sample snort alert:

> [**] [1:1748:8] FTP command overflow attempt [**]
> [Classification: Generic Protocol Command Decode] [Priority: 3]
> 11/24-14:02:32.735830 192.168.1.4:3247 -> 192.168.1.1:21
> TCP TTL:128 TOS:0x0 ID:20571 IpLen:20 DgmLen:358 DF
> ***AP*** Seq: 0x1C5D5B76 Ack: 0x681EACAD Win: 0x4470 TcpLen: 20
> [Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0606][Xref =>
> http://www.securityfocus.com/bid/4638]

Visit the URLs contained in it. What is the purpose of an "Xref" in a snort alert?

4. Here is a rule:

alert tcp $HOME_NET 23 -> $EXTERNAL_NET any (msg:"TELNET login incorrect"; content:"Login incorrect";)

Explain it. Under the circumstances the rule represents, who is doing what? where? State precisely to which packets the rule applies, and what is the resulting action when such packets are seen. Explain the difference between the roles played by the two embedded strings "TELNET login incorrect" (what's that? why is it there? what does it do?) versus "Login incorrect" (why is it there? what does it do?).