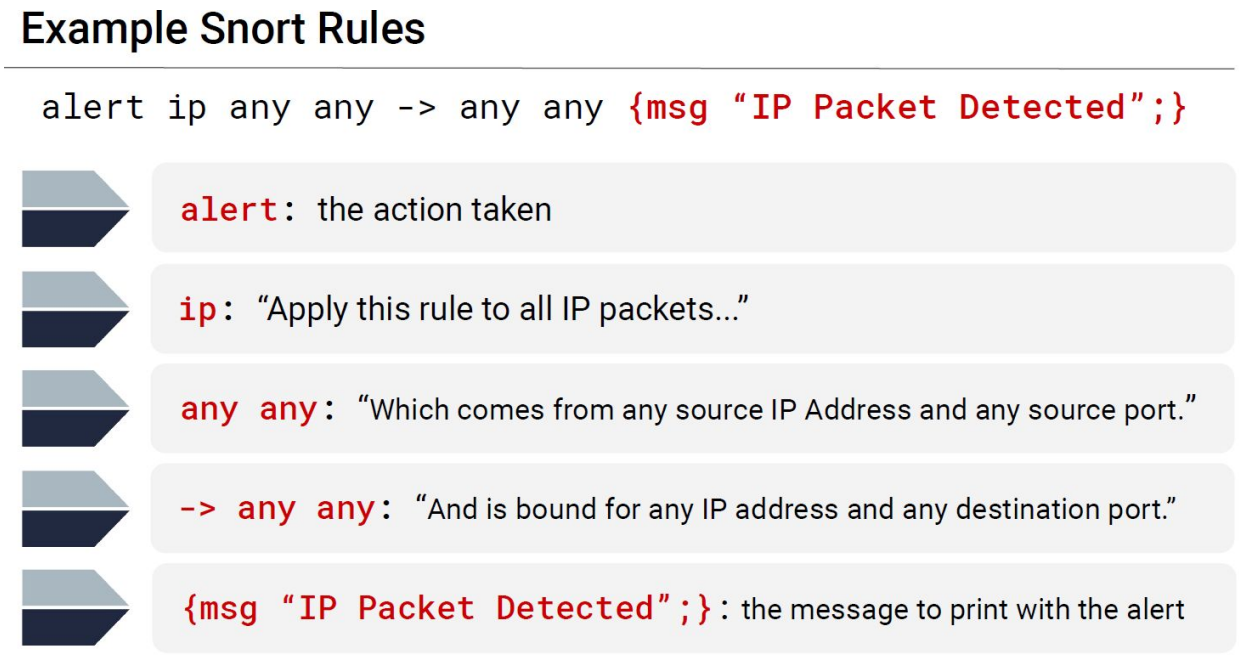
Terminal Commands Summary, Part III (Lesson 9)

This lesson introduced Intrusion Detection systems that can be configured amongst Firewalls and routers between the internet and users/admins in a network.

Snort - analyzes traffic through a series of analyzers. For more information, see Linux 2, Day 1 slides. Example Snort Rule:



To launch Snort in packet-capture, verbose mode (note path to conf file):

`snort -bv -c /etc/snort/etc/snort.conf`

Stop Snort with Ctrl+C

The `-q` flag starts Snort in quiet mode. This prevents it from logging packets to the screen:

sudo snort -bq -c /etc/snort/etc/snort.conf

-A flag for full Alerts

Launch Snort in quiet mode with full alerts, and put it in the background with `&`. Run `ps` to ensure that Snort is running.

`sudo snort -A full -bq -c /etc/snort/etc/snort.conf & && ps`

Can run `head alert` to see first alerts; or `tail alert` to see most recent alerts

/etc/snort/rules : directory that contains snort rules

A more complicated snort rule:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any
(msg:"PROTOCOL-ICMP PING Unix"; itype:8; content:"|10 11 12
13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F|"; depth:32;
metadata:ruleset community; classtype:misc-activity;
sid:366; rev:11;)
```

Breaking it down:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (
```

Action: alert and log packets that trigger this rule
Protocol: apply rule only to ICMP traffic
Source/Destination addresses: any packet from outside the local subnet (any port) into the local network (any port) that matches the rule will fire an alert
Rule Options: Everything else are options - what the rule looks for, how to print output, etc.

1.) msg: The string to log when the rule is triggered
2.) itype: Check for a specific type of ICMP packet (Type 8 is an echo packet)
3.) content: Data to look for inside the packet
4.) depth: specifies how many bytes into the packet Snort should look for content
5.) metadata: administrative information about the rule. In this case, a community rule
6.) classtype: specifies what kind of network activity this is
7.) sid: the id of the rule
8.) rev: The revision id. It is essentially a version of the rule

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (
    msg:"SQL PK-CMS SQL injection attempt";
    flow:to_server,established;
    content:"/default.asp?"; fast_pattern;
    nocase; http_uri;
    content:"pagina="; distance:0; http_uri; pcre:"/pagina=[^&]*\x27/Ui";
    metadata:service http;

reference:url,github.com/BuddhaLabs/PacketStorm-Exploits/blob/master/1309-exploits/pkcms-sql.txt;
    classtype:web-application-attack;
    sid:32768;
    rev:1;)
```

This exploit watches for the `pkcms-sql` exploit.
This exploit is delivered via the GET query string, through the parameter `pagina`.

Write a rule that detects telnet traffic from the public Internet to the local subnet.

alert tcp $EXTERNAL_NET any -> any 23 (msg:"Telnet packet detected!";sid:2000000;)

Write a rule that detects an attacker running a tcp scan on any port.

alert icmp any any -> any any (msg: "TCP scan";sid:10000004;)

Metacharacters

| (pipe) - two or more commands such that the output of one serves as the input to the next

`ls /usr/bin | wc -l`: List the contents of `/usr/bin`. Count how many files it contains (wc -l)

`ps aux | grep bash | wc -l` : Print running processes->find out the string `bash->count how many shells are running

* (wildcard) i.e., How many programs in `/usr/bin` start with the letter `a`

`ls /usr/bin/a* | wc -l`

Find all files that end with `.png` inside of your `~/Documents` directory

`find ~/Documents -type f -iname '*.png'`

Print names of everything in the current directory - `echo *`

Print every file in the current directory - `ls *`

Expansion

Variables - VAR=VALUE syntax. Reference using $. I.e., NAME="Bruce" … echo $Bruce

Use export to create a variable that persists across all shells, terminal windows/logins

I.e., `export PATH=$PATH:$(pwd)`

~ - To reference home directory. I.e, echo ~

file - tells you the type of a file

which - tells you the absolute path for a command

Figure out what kind of file `ls` and `cp` are

`file $(which ls)` and `file $(which cp)`

&& (And) - Concatenates commands. Executes until one fails

cat /etc/shade && echo "Shadow" - will just error since part before && fails

echo "Shadow" && cat /etc/shade - Shadow echoes then error for part after &&

|| (Or) - Concatenates commands. Executes until one succeeds

cat /etc/shade || echo "Shadow" - Part before || fails, then continues, after || succeeds

echo "Shadow" || cat /etc/shade - Shadow echoes then error for part after &&

; (Semicolon) - Concatenates commands. Executes all. In the above examples, both commands execute regardless of success or failure.

I/O Data Streams

Data Stream - a way to describe channels of data as they are processed and moved through a system

stdin - used to stream input data

stdout - used to stream output data

stderr - used to stream error data

`cat < /etc/passwd` vs `cat /etc/passwd`
No difference really - they both send `etc/passwd` to cat as standard input. Using `<` is explicit

`cat << EOF` - to wait for text (called a heredoc) until EOF
cat << EOF > my_file.txt - same but save standard output to my_file.txt

Throw Error Demo

```
file $(find . -iname *.txt 2> /dev/null) > ~/Desktop/text\ files ; tail ~/Desktop/text\ files
```

1.) $() for command expansion
2.) . To reference your current location
3.) * expands to any number of characters
4.) 2> redirects stderr to dev/null in order to hide errors
5.) > redirects stdout to new file
6.) ~ expands to the home path of the user
7.) \ escapes the space in the file name
8.) ; runs the tail command while the find command is still running

For Loop Syntax

```
for Name in “Moe” “Larry” “Curly”
do
        echo $Name
done
```

- Start all scripts with: #!/bin/bash

Read: to read in console arguments as in: read arg1. Can reference value as $arg1 in script
First argument in script ($0) is the name of the script.

Sample script to ping targets on a subnet with a user supplied routing prefix. Uses seq in a looping mechanism:

```
#!/bin/bash

# Get IP prefix from the user
echo -n "Enter routing prefix: > "
read PREFIX
```

```
# Ping IPs on the subnet
echo "Scanning $PREFIX.0/24..."
for i in $(seq 1 255)
do
        # Append counter to user-supplied prefix for all ips on subnet
        TARGET=$PREFIX.$i

        # output Target to screen
        echo $TARGET

        # ping the target once, hide stderr and stdout; echo Target  Up or Target Down instead
        ping -c 1 $TARGET 2>&1 > /dev/null && echo "$TARGET is UP!" ||  echo "$TARGET is
DOWN"
done
```