Integration of Snort NIDS with Splunk for Threat Hunting and Penetration Testing

PREPARED FOR

Professor Ali Hadi

SEC350-02

PREPARED BY

Caitlin Allen & Doug Kapinos

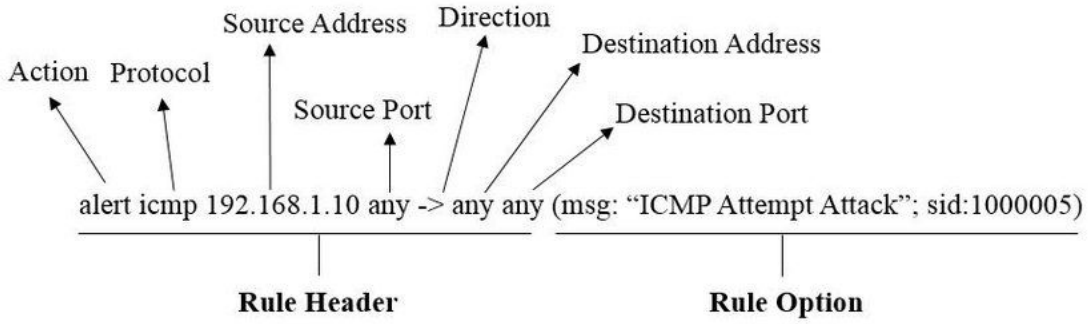
Your students

Project Goals and Scope

In an environment set up from scratch, we integrated the network intrusion detection system (NIDS) Snort with our security information and event management (SIEM) solution Splunk. The environment was set up to meet all requirements for segmentation, firewalling, and logging (which will be highlighted in the documentation).

In a working environment, your IDS system, whether it be Snort, Zeek/Bro, Suricata, or Solarwinds, should be right behind the firewall, with the firewall placed in front as the first line of defense. For simplicity, we located ours within the DMZ. Now you may find that Endpoint Detection and Response (EDR) tools are more commonly used in commercial organizations using pricey solutions from top companies like Crowdstrike's Falcon or Elastic's Endgame. But for a smaller organization or a private network like our environment, financially it is not feasible.

Snort is a free NIDS, with the option to pay for different rule sets to use in our firewall. Snort also provides sniffing and packet logging modules to use as well, making it a well-rounded packet analysis tool. We will be using the Community Rules and designing a few of our own local rules. Snort rule sets are easy once you get the syntax down and can understand the rule structure.



This is the general syntax you want to follow. For any additions to the syntax, they will be explained.

Log output can be formatted in a few different ways, like:

- alert_syslog
- alert_fast
- alert_full
- alert_unisock

There are a few different other ways to format them, but we will be using alert_syslog. The alert_syslog format ships logs to the syslog, you can specify the logging facility and priority within the Snort config file and ship these logs to your SIEM solution.

Our penetration testing platform is Mutillidae, is an open-source deliberately vulnerable web-application that allows upcoming pen testers or web app security enthusiasts to practice exploits. This web app is installed using the LAMP stack, composed of four different open-source components:

- Linux
- Apache
- MySQL
- PHP/Perl/Python

Mutillidae is easy to use and allows the user to toggle levels of security, gives hints, and offers OWASP resources on different exploits.

Figure 1: Mutillidae home screen on our Ubuntu Web workstation

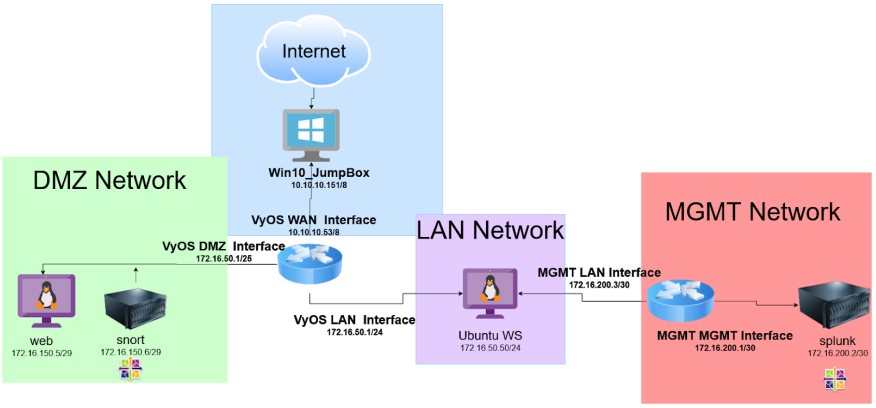


Figure 2: Network topology of our environment

1. Setting up Snort on CentOS 7

In these instructions, we will assume your segmented network already exists. Our focus is specifically on our snort server and the Ubuntu web workstation. Your SIEM solution for integration should be properly configured to ingest logs forwarded to it.

We need to install from the source, Snort ran into issues with installing from yum. This documentation is for the current version of Snort available, the download link for a more current version later is available on the Snort website under Binaries.

1. Installing from source.

```
sudo yum install
https://www.snort.org/downloads/snort/daq-2.0.6-1.centos7.x86_64.rpm
sudo yum install
https://www.snort.org/downloads/snort/snort-2.9.16-1.centos7.x86_64.
rpm
```

2. Create folder structure for Snort. It is best to do this now, issues arose if it was done after other steps.

```
mkdir -p /etc/snort/rules
mkdir /var/log/snort
mkdir /usr/local/lib/snort_dynamicrules
```

3. Set directory permissions

```
chmod -R 5775 /etc/snort
chmod -R 5775 /var/log/snort
chmod -R 5775 /usr/local/lib/snort_dynamicrules
chmod -R 5775 /usr/local/lib/snort_dynamicrules
chown -R snort:snort /var/log/snort
chown -R snort:snort /usr/local/lib/snort_dynamicrules
```

4. Create new rule files, this is needed so the configuration file is able to execute properly.

```
touch /etc/snort/rule/white_list.rules
touch /etc/snort/rules/black_list.rules
touch /etc/snort/rule/local.rules
```

5. Configure Snort to run in NIDS mode

```
sudo ldconfig
```

```
sudo ln -s /usr/local/bin/snort /usr/sbin/snort
```

6. Set up the Community rules, these are user made and important in making sure Snort runs properly

```
wget https://www.snort.org/rules/community -O ~/community.tar.gz
sudo tar -xvf ~/community.tar.gz -C ~/
sudo cp ~/community-rules/* /etc/snort/rules
sudo sed -i 's/include \$RULE\_PATH/#include \$RULE\_PATH/'
/etc/snort/snort.conf
```

7. Configure the network and rule sets by opening the Snort configuration file.

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

8. Make the changes listed below

   In Vim, you can use / to search for lines! This is a long configuration file so be sure to make sure you made these changes.

   Add the network address and mask to this line

```
# Setup the network addresses you are protecting
ipvar HOME_NET Network_To_BeProtected/XX
```

```
# Set up the external network addresses. Leave as "any" in most
situations
ipvar EXTERNAL_NET !$HOME_NET
```

```
# Path to your rules files (this can be a relative path)
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules
```

```
# Set the absolute path appropriately
var WHITE_LIST_PATH /etc/snort/rules
var BLACK_LIST_PATH /etc/snort/rules
```

```
#syslog
Output alert_syslog host=IP_OF_SYSLOG_BOX:514 LOG_AUTH LOG_ALERT
```

```
# unified2
# Recommended for most installs
output unified2: filename snort.log, limit 128
```

Uncomment the following lines

```
include $RULE_PATH/local.rules
```

```
include $RULE_PATH/community.rules
```

9. Validate the Snort settings

```
sudo snort -T -c /etc/snort/snort.conf
```

10. If you get an error, run this command and retry to validate the settings.

```
ln -s /usr/lib64/libdnet.so.1.0.1 /usr/lib64/libdnet.1
```

When Snort is successfully installed and initialized, you should see this.

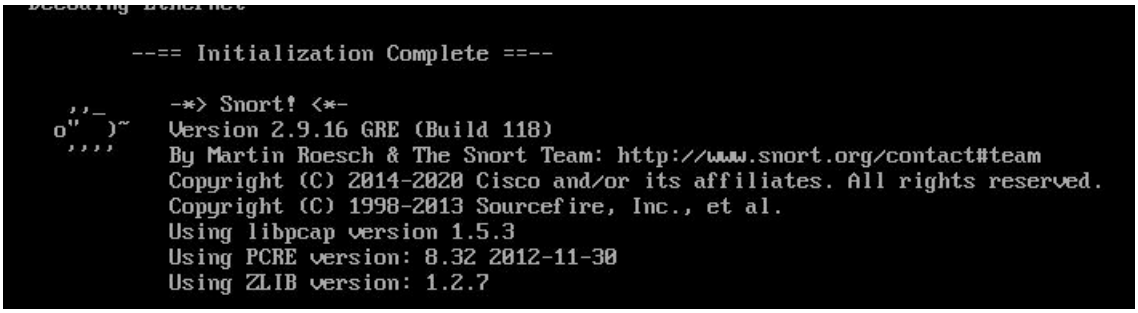


Figure 3: Snort successfully stood up in CentOS 7

11. You now can test the configuration by setting a basic rule to alert ICMP connections. This can also be tweaked to alert for potential nmap attempts since nmap utilizes ICMP for OS Fingerprinting, service detection, and network scanning.

    Open the rules up

```
sudo vi /etc/snort/rules/local.rules
```

12. Add this to the file

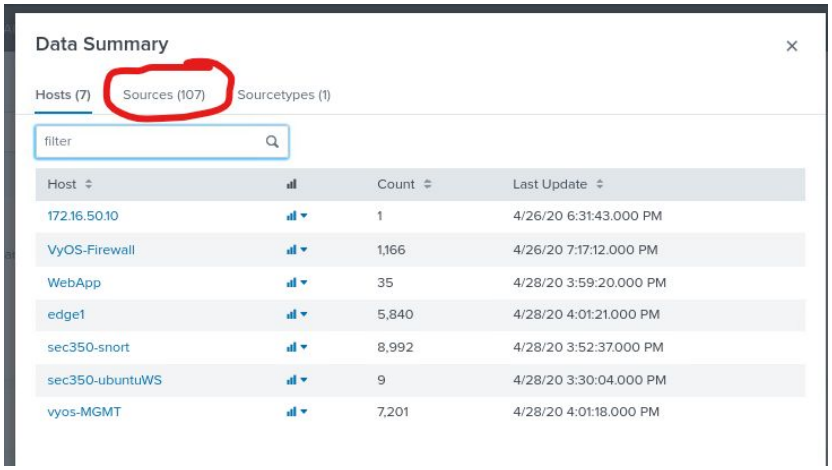
```
alert icmp any any -> $HOME_NET any (msg:"ICMP test"; sid:10000001;
rev:001;)
```

13. Start Snort and ping the snort host from your web box. You should see ICMP test alerts roll in on the screen.

```
sudo snort -c /etc/snort/snort.conf -i [Interface]
```

2. Snort logs on Splunk

Open up the search application on your Splunk host. You can view the logs by host, however, it is easier to search by source. Look for the snortd.log. You can update by latest logs in this to find it easier.



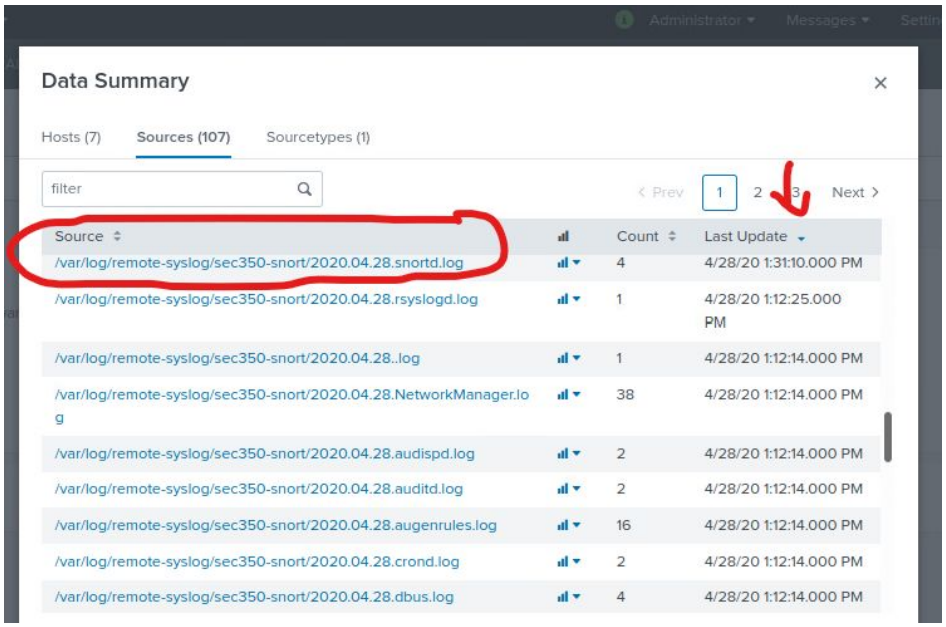


Figure 4: Splunk snortd.logs and where to find them in the search app

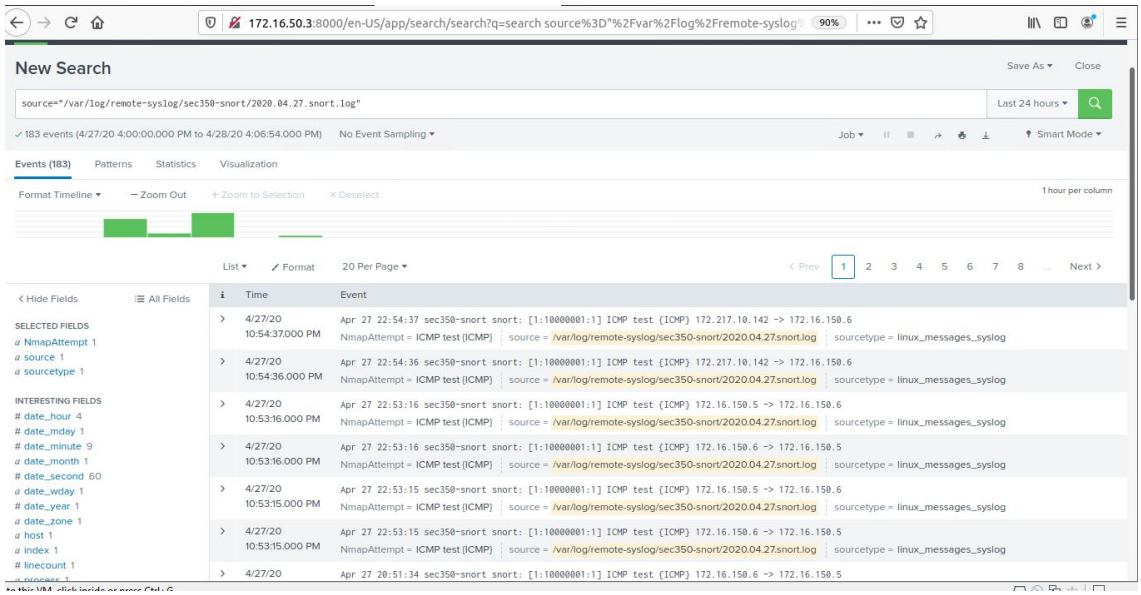


Figure 5: ICMP test logs (Field extraction done for Dashboards)

3. Setting Up Mutillidae

On the snort host, you will be installing the following packages.

1. httpd
2. mariadb-server
3. mariadb
4. php
5. php-mysql
6. php-pear
7. php-pear-db
8. php-mbstring
9. git

This is a CentOS box so yum is how you should be downloading these packages!

Start the httpd service after installation and open port 80 with firewall-cmd

We need to do a MySQL secure installation.

Run the following

```
sudo mysql_secure_installation
```

This will prompt you for your root password then ask you to change it. You do not need to change your root password if you do not want to.

For the following questions after the password prompt, make sure to allow remote access

Now, run the following to grab Mutillidae off Github

```
sudo git https://github.com/webpwnized/mutillidae
```

Copy the Mutillidae folder into /var/www/html

Then navigate to http://**LOCALHOST_IP**/mutillidae

If successful, you should see the following

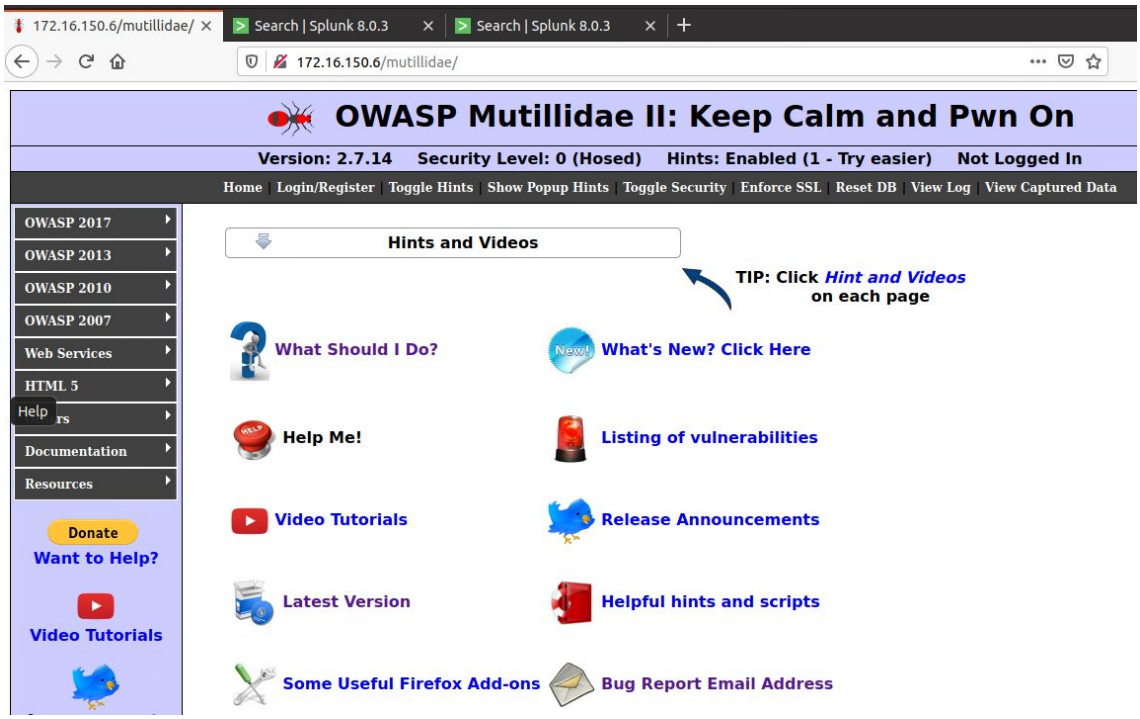


Figure 6: Mutillidae

4. Integrating Snort with Mutillidae

Since Snort is a traffic analyzer, let's set some rules to pick up potential attacks from Mutillidae.

Open the local.rules file again.

Lets add some basic rules to detect SQL Injections, Command Injections, and Cross-Site Scripting attacks.

```
#ICMP Connection rule -> Testing purposes
alert icmp any any -> $HOME_NET any (msg:"ICMP test"; sid:10000001; rev:001;)

#SQL Injection rules
alert tcp any any -> any 80 (msg: "SQL Injection Detected"; content: "%27" ; nocase; sid:100000011;
)
alert tcp any any -> any 80 (msg: "SQL Injection Detected"; content: "22" ; nocase; sid:100000012; )

#XSS Rules
alert tcp any any -> any 80 (msg: "Cross-Site Scripting Attack Detected"; content: "%3Cscript%3E" ;
nocase; sid:100000013; )
alert tcp any any -> any 80 (msg: "Cross-Site Scripting Attack Detected"; content: "<script>" ; noca
se; sid:100000014; )

#Command Injection Attempts
alert tcp any any -> any 80 (msg: "Command Injection Detected"; content: "cat /etc/passwd" ; nocase;
 sid:1000000015; )
alert tcp any any -> any 80 (msg: "Command Injection Detected"; content: "cat /etc/shadow" ; nocase;
 sid:100000016; )
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
                                                                    5,1           All
[0] 0:sec350final@sec350-snort:/var/log*                    "sec350-snort" 02:38 28-Apr-20
```

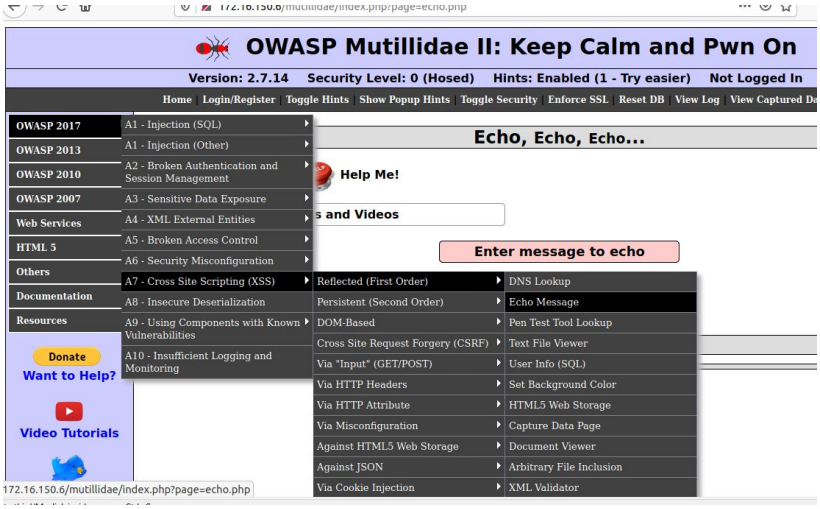
Figure 7: Snort local.rules

Our Snort rules are pretty basic, but here is a good place to find some more advanced SQL and XSS related rules.

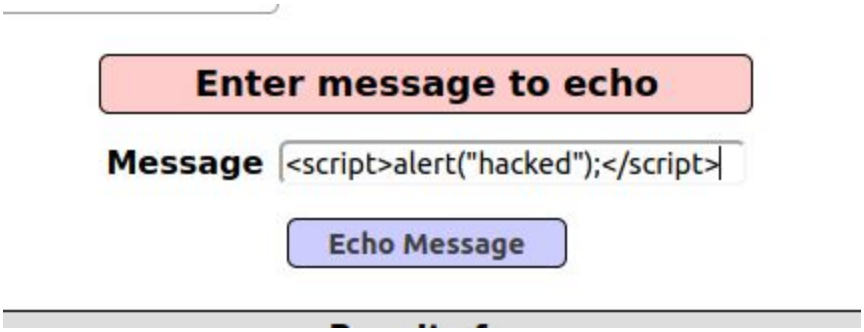
Lets test our Cross-Site Scripting rule.

In Mutillidae:

1. OWASP 2017 > Cross Site Scripting (XSS) > Reflected (First Order) > Echo Message

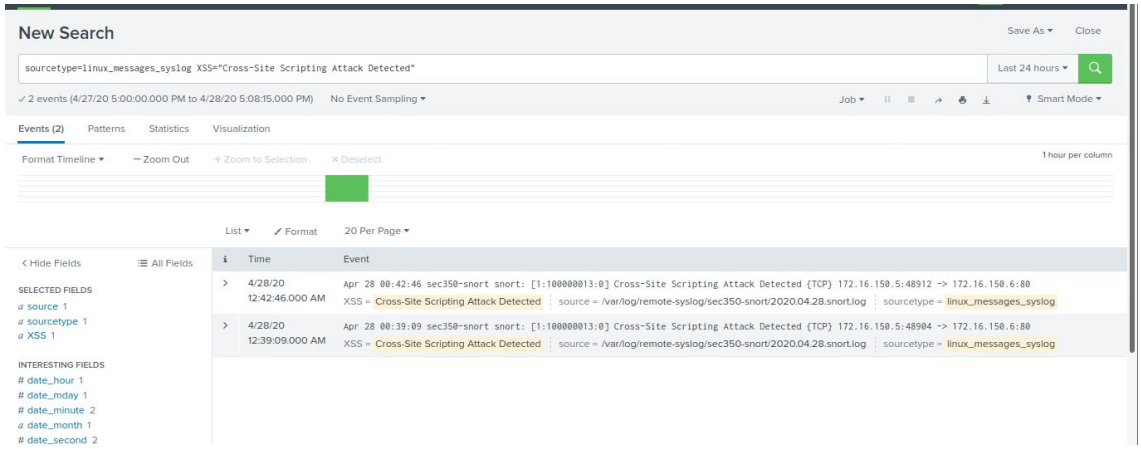


2. Enter the Following



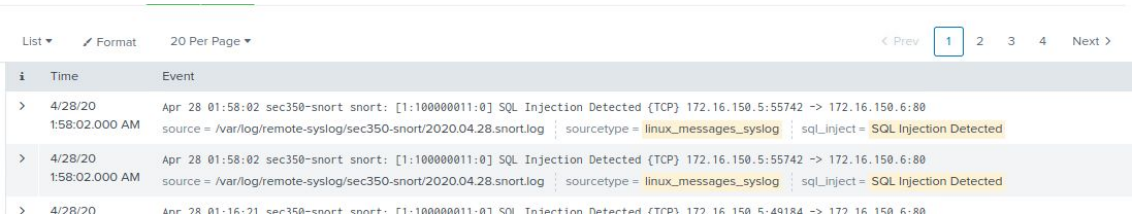
3. An alert should pop up if successful. Lets go to Splunk

4. If successful, logs should roll through.



Try other methods to ping off your command injection alert and SQL injection alert. Heres some hints

Try to get some logs like this and we can begin to make dashboards



5. Dashboard Building

Every good SOC analyst wants a comprehensive dashboard they can open Splunk right up to.

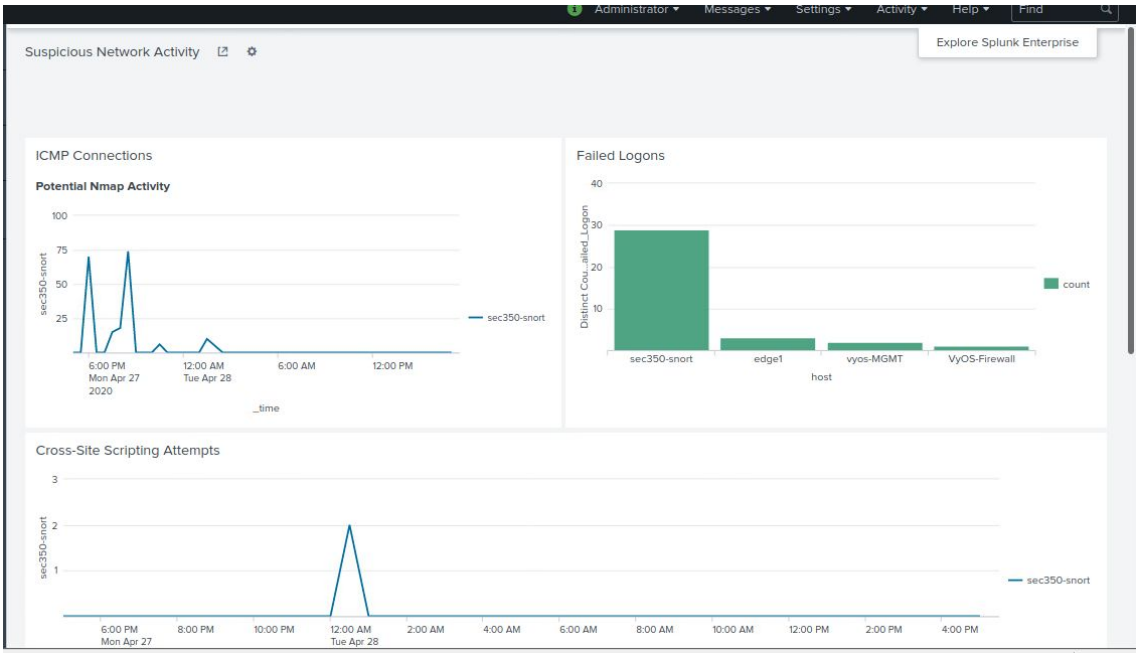


Figure 15: Home dashboard

In the search app, open up the Dashboard module.

Lets make a dashboard like this one, but for our Snort logs.

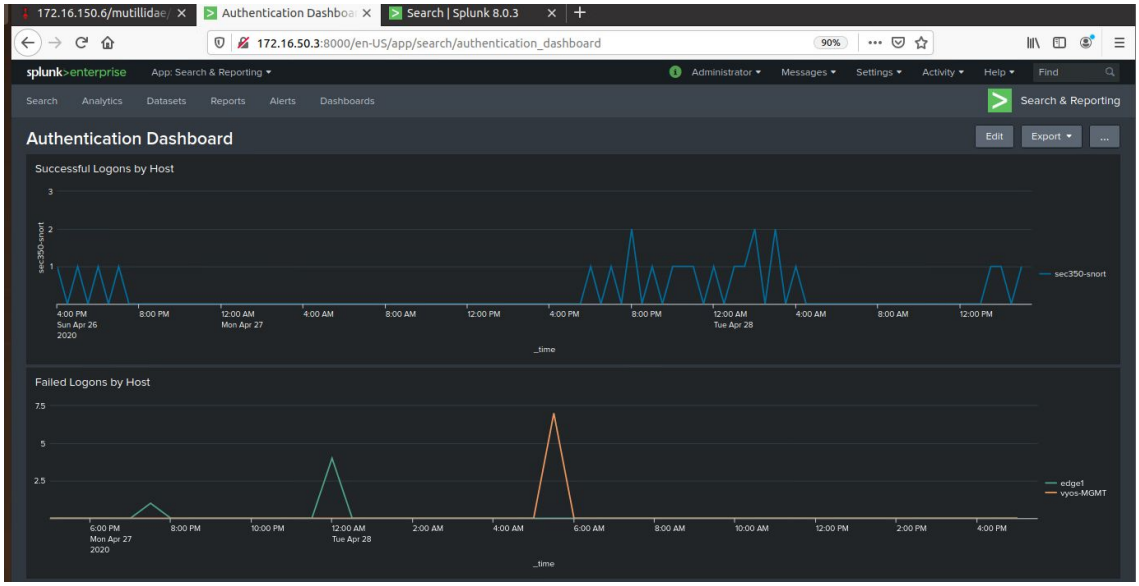


Figure 16: Authentication dashboard

Lets make a line graph of Cross-Site Scripting Attempts

Create a line graph dashboard panel, for the search, we'll want to see what hosts are spiking in activity.

Enter this for your search using your snort.log

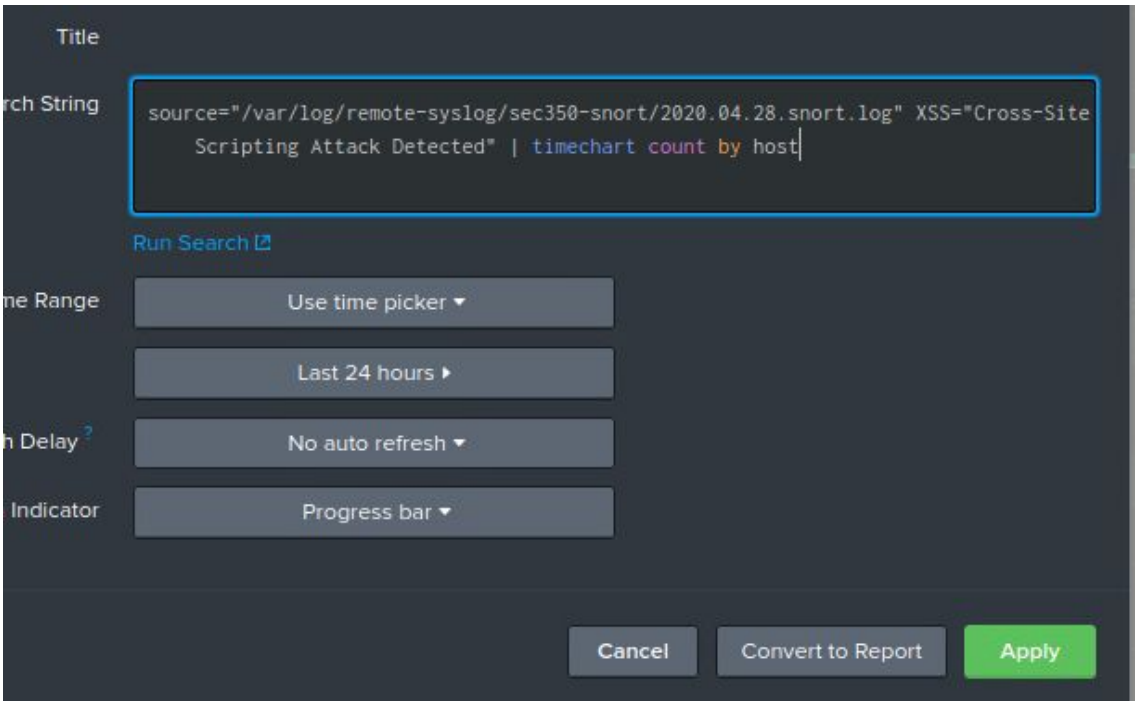


Figure 17: Search syntax for panels

You want to pipe your results to timechart to show the timing of the attacks, which would be needed for incident response to generate a timeline and identify when an attacker made attempts, you will then count by host to count each time a host has been breached.

Figure this out for the other attacks and even add a few other panels, we included nmap attempts for ICMP tests and failed logons by host in case of brute force. To make this easier, use field extraction to extract the exact log messages from Snort you are looking for.

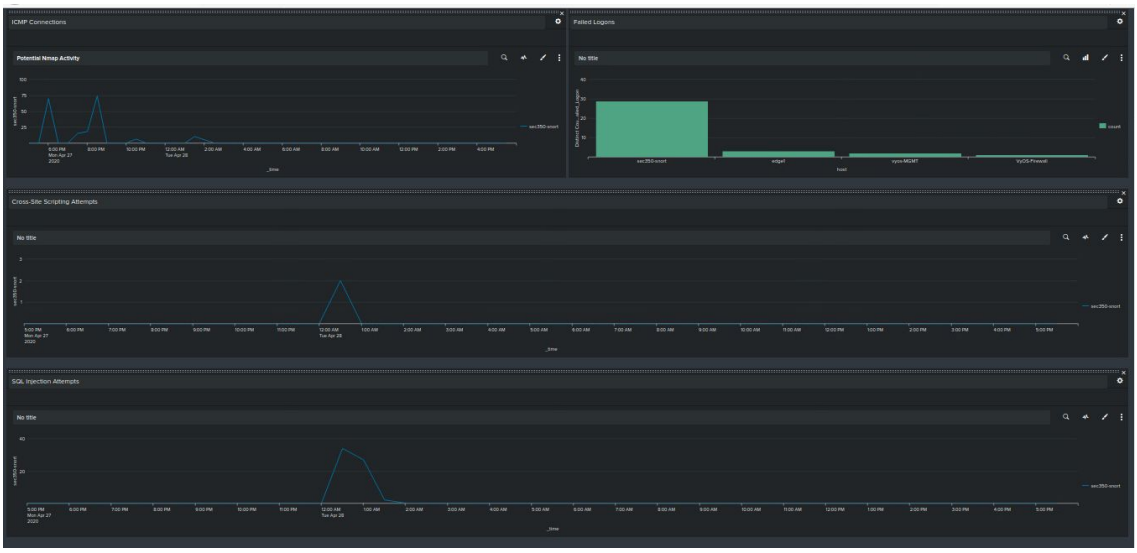


Figure 18: Final dashboard of Snort logs and failed logons with ICMP connections

You can add this to your home page if you like!

7. Troubleshooting & Issues

Our environment was built on Caitlin's HP Proliant DL380 Gen 7 server with ESXi hypervisor. Different network segments were created within ESXi to segment the network and configure the firewalls.

Each VM had to be loaded as an ISO then deployed. One issue we discovered early on was with the unstable version of VyOS we were using, since stable current versions cost money, anytime Caitlin shut down the server, our VyOS settings were wiped from the entire VM. Committing to the configuration file did not matter, VyOS essentially had volatile memory. To solve this, anytime any work was done on VyOS, we snapshotted the machine and were able to overcome this by restoring from the snapshot.

Another issue was Snort's network placement. It was not intended to be set up as a host but as a NIDS for the DMZ. However, just due to configuration issues and realizing that it would not intercept until too late, we decided to put Mutillidae on snort and log directly from the box. To solve this, we would have had to do major network reconfigurations.

8. Conclusion

Snort is a fun packet analyzer with many capabilities! Using the provided link and Snort documentation, building rules to integrate with Splunk

You can use simple rules to get started and use regex, encoding, etc. to get more precise in what your NIDS will detect and ship to Splunk.

Our environment was built on Caitlin's HP Proliant DL380 Gen 7 server with ESXi hypervisor. Different network segments were created within ESXi to segment the network and configure the firewalls.