Snort Intrusion Detection System

Tony Perez
SUNY Polytechnic Institute
100 Seymour Rd
Utica, New York
pereztr@sunyit.edu

Jess Yanarella
SUNY Polytechnic Institute
100 Seymour Rd
Utica, New York
yanarej@sunyit.edu

ABSTRACT

This project will show how we setup an Intrusion Detection System using open source software called Snort. We have setup a Linux box in the CS Department Network. It is on a physical machine running CentOS minimal and the necessary packages for Snort to run properly. Besides showing Snort, we have implemented other things to show how the traffic being sniffed can be viewed in a user friendly graph online. This uses the features of Ruby on Rails Application known as Snorby. In this project, we want to create all sorts of traffic that can be visible in the command line logs and the graphical interface. We implemented the Snort rules that come pre-configured with the installation. We will also test other rules that we created on our own to generate alerts for certain traffic. We created a virtual machine on another laptop to use a program called Pytbull. This allows for traffic to be automatically generated.

Figure 1: Snort Logo Trademark of Sourcefire
Source: www.rivy.org

Categories and Subject Descriptors

C.5 [**Information Systems Organization**]: Computer System Implementation; C.5.5 [**Networking & Cyber Security**]: Monitoring

Keywords

Linux, Snort, CentOS, Intrusion Detection System

1. INTRODUCTION

Many businesses and organizations pay for top dollar equipment to detect threats in their network. Snort is a free open source program that can be installed and configured to the users' preference on different machines. It has many features and their rules are very reliable. A lot of owners may not be aware of the risks their network faces to outside attacks. Snort can help monitor traffic and output all results in its' log file. Everything that is going on can also be visually shown by using Snorby. This ruby on the rails application is a user friendly GUI interface accessible from the Internet.

All across the world today, the field of cyber security is growing rapidly. The amount of hacks and threats made against big organizations is increasing everyday. Recently, two well known companies, Sony and Anthem Health Insurance were breached in a successful attack on them. They both lost a lot of confidential data to the attackers, which some can become public information at a cost. More companies need to be aware of everything that is occurring on their network and a specified team should be assigned to this task. The biggest problem here is that many people probably don't even know about this cyber war we are apart of. Equipment can be bought to help protect private data, but open source software like Snort allows users free will to configure everything. This project focuses on mainly detected any issues and services not preventing them. Farther into the software, users can get into protecting their system from threats.

Snort used to be owned by Sourcefire but now Cisco bought them out. Since Cisco now owns Snort, certain software related to Snort is becoming extinct as it has not been updated in over two years. Related programs that helped Snort function in more ways are Pulled Pork and Barnyard2. "Pulled Pork was helpful because it helped download and copy pattern files to the right location. Barnyard2 was a tool that could read the log files associated with Snort and convert them to be sent to a database." [4]

2. RELATED WORK

In previous lab we have got involved with a program called Wireshark which captures all network packets.

We have not done many labs involving intrusion detection systems, but we have learned about them extensively. After looking more into IDS, we could see that users need them to check if anyone is attempting to break through firewalls.

3. SETTING UP SNORT

For our setup we needed:

- Club Space on the NCS Network
- CentOS Linux Box
- Cisco Catalyst 3550 Switch
- Virutal Machine with Pytbull installed

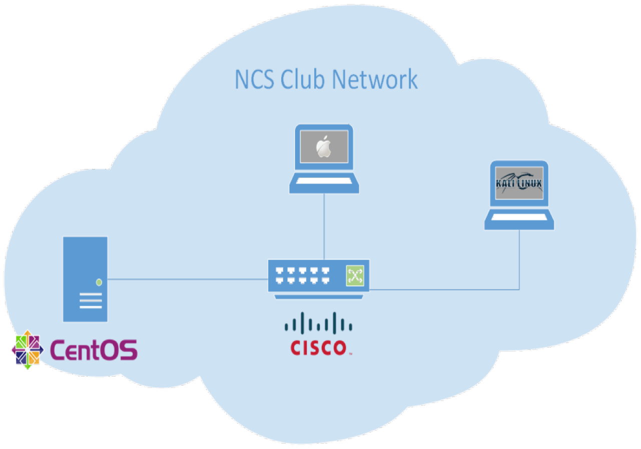


Figure 2: NCS Club Room Environment

4. INTRUSION DETECTION SYSTEMS

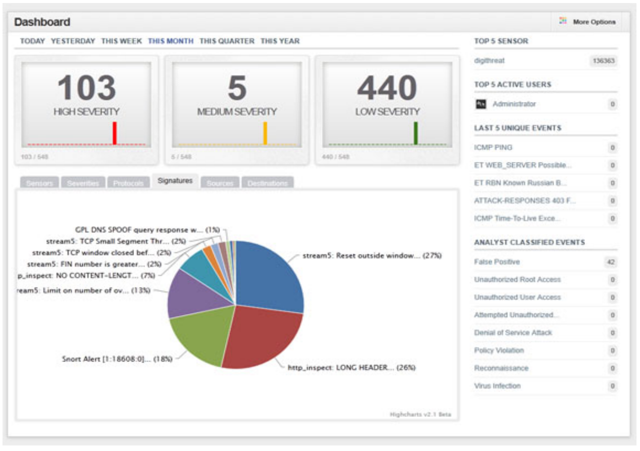
Intrusion detection systems help users track and view all activity on their network. Many people may not be aware of what's going on and Snort is the perfect solution. This software is great for all users to learn and companies to involve their teams with. There are two types of intrusion detection systems, network based and host based. "Network based systems monitor traffic on an entire network."[3] Each machine has their own network interface card which can function in two different modes, promiscuous and normal. "The fact that even though a packet arrives at the host it was sent to, it still has another line of defense that is setup behind the firewall and this is known as a host based intrusion detection system."[3]

5. SNORT INSTALL

To setup Snort on CentOS we had to compile from source and make sure the right permissions were set. The Snort daemon will run in the background and so it must not run as root but as its own user, which we named `snort`. We compiled snort and then went on to add the script used to start, stop, and restart the snort daemon. After installing we had to get the rules we wanted to use. After testing the basic configurations and basic rules to see if snort was able to pick something we went on and registered on Snort's website to get a set of rules that have been tested and from Sourcefire themselves. These rules were used to detect specific type of packets like certain types of malware, exploits, ad-ware, blacklisted IPs, and more. The set of rule were extensive but well documented and so we could understand what the rule is checking and how. The snort installation process and rule setup was smooth especially when using the provides rules and configuration once you register.

6. INSTALLING SNORBY

Snorby is a graphical interface that connects with your Snort Box to be accessed online. This is GUI display that neatly organizes all the alerts that were set off. It groups everything that the system detected and groups them in different categories; low, medium, or high severity. We were able to install Snorby, but it used an outdated version of Ruby and did not save any alerts into the database. We could access the interface, but it didn't show any of our results. For the final presentation, we decided not to include Snorby because it was not working properly to be shown to the class. We decided that using the command line log will be sufficient enough to display all the alerts we set up by running Armitage.


Dashboard
More Options
TODAY
YESTERDAY
THIS WEEK
THIS MONTH
THIS QUARTER
THIS YEAR
103
HIGH SEVERITY
103 / 548
5
MEDIUM SEVERITY
5 / 548
440
LOW SEVERITY
440 / 548
Sensors
Severities
Protocols
Signatures
Sources
Destinations
GPL DNS SPOOF query response w... (1%)
stream5: TCP Small Segment Thr... (2%)
stream5: TCP window closed bef... (2%)
stream5: FIN number is greater... (2%)
p_inspect: NO CONTENT-LENGT... (7%)
ream5: Limit on number of ov... (13%)
stream5: Reset outside window... (27%)
Snort Alert [1:18608:0]... (18%)
http_inspect: LONG HEADER... (26%)
Highcharts v2.1 Beta

TOP 5 SENSOR

digithreat	136363

TOP 5 ACTIVE USERS

Administrator	0

LAST 5 UNIQUE EVENTS

ICMP PING	0
ET WEB_SERVER Possible...	0
ET RBN Known Russian B...	0
ATTACK-RESPONSES 403 F...	0
ICMP Time-To-Live Exce...	0

ANALYST CLASSIFIED EVENTS

False Positive	42
Unauthorized Root Access	0
Unauthorized User Access	0
Attempted Unauthorized...	0
Denial of Service Attack	0
Policy Violation	0
Reconnaissance	0
Virus Infection	0

Figure 3: Snorby Graphical Interface
Source: www.digitalthreat.net

Issues:

- Wrong Ruby Version (2.2)
- Could not install older Ruby version from source
- Permissions were not set correctly
- Problem using Phusion Passenger

Solutions:

- Installed Ruby 1.9.3
- Used rvm
- Mysql database the correct user and Snorby files in web directory
- Remove Phusion Passenger

The snorby setup was not smooth only because it was using an outdated ruby version and so many of the gems it used were not compatible with the lastest ruby. When using the older version of ruby and was up and running it did not want to pull data from the data base. This was an issue with

Barnyard, which is used to put the alerts and rules into a database and format the output as well. Barnyard is another third-party application that was used to make Snort more efficient and cause it to not miss any traffic. Snorby needs a database to pull data from and so we could not show any version of Snorby working.

7. SWITCH SETUP

For our switch, we used a basic configuration setup so we can connect multiple devices together. The only thing we really needed was a span port to be setup on the Internet side. This allowed for the hosts to send everything through the switch and the span port to mirror the traffic to our Snort Box. In the diagram we have our hosts sending traffic we generated through the Internet, which is on the span port. It then mirrors everything to the Snort Box and it catches all the malicious attempts.

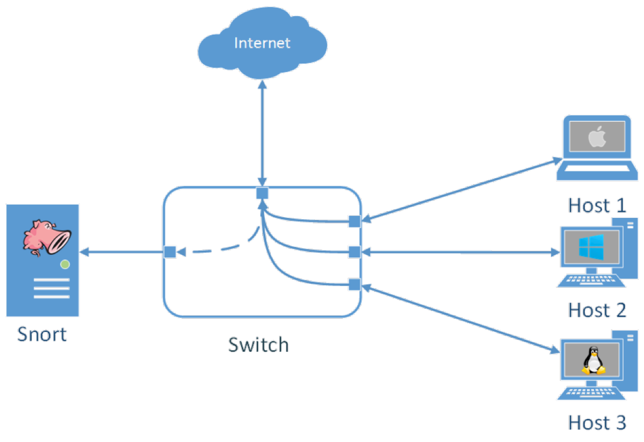


Figure 4: Span Port

8. TESTING SNORT

In our project, we used Snort for testing purposes on the NCS network. We are restricted to what we can test because we are doing it on the actual network other students use everyday. Snort can detect many services, but a lot of them include malicious activity. Within the snort installation, there are rules files located at `/etc/snort/rules/`. These rules are different alerts that will be triggered when they detect specific activity. Snort comes pre-installed with rules and they need to be set as active by uncommenting them in the configuration. There are many rules already written for users to figure out and decide which ones they want to use. The category of the rules are well documented on Snort's website and they include[2]:

- app-detect.rules
- blacklist.rules
- browser-ie.rules
- malware-backdoor
- os-windows
- policy-social
- pua-adware (PUA: Potentially Unwanted Application)

For our purposes, we want to use all of them so we can generate as much alerts as possible to demonstrate the project. We came across some problems on how we can set these alerts off with malicious attempts on the network. Once it was setup, we could start the log with the command `tail -f /var/log/snort/alerts`. In this log, it outputs everything that it detected from the rules. We decided to create our own simple rule for ICMP and saw a alerts going off for this service as soon as we pinged our Snort Box.

```
alert ip any any -> any any ( msg:  "ICMP: SOMEONE IS
PINGING ME!!!!"; sid:10001; )
```

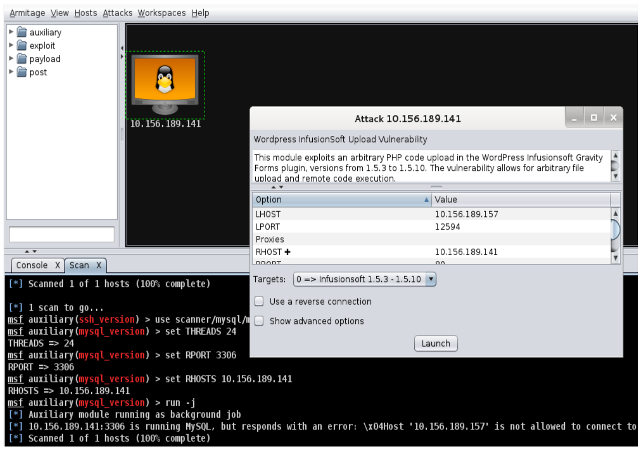
Snotr's rules go more in depth and to figure out which rules worked we used the most we could and tested them to see which alerts go off. We were having a hard time figuring out how to test a large portion of the rules and did not realized that there were pcap files available online which contain malicious traffic we could replay. Another great way to test an IDS is to setup it up during a cyber defense competition and see how much malicious traffic is going on during the event. We decided to just find a tool that generated malicious traffic for us.

9. PYTBULL

We have successfully installed Snort and got it up and running smoothly. Now that it is working, we needed a way to generate a bunch of traffic for to demonstrate. After a lot of research, we came across the software Pytbull; an IDS/IPS testing framework. This tool would of been great to use because it checks with the Snort configuration files and generates traffic based off the rules. We came across a lot of problems with this software because it is out of date and the services it uses no longer support it. We tried different ways to install it, but each time we got the same error. The error had to do with Python being a newer version then what Pytbull was built on. We first installed an Ubuntu VM and downloaded the software on there. We tried to run it, but got an index error so we thought let's just remove Python and download the appropriate version 2.6.5. This did not give us any luck either since Pytbull requires other services with Pytbull like scrappy. When we prompted to install those packages, it would just update Python back to the newest version. Next, we saw that Pytbull was pre-installed on Backtrack 5 R3 so we installed a VM of that. This already had the configuration files and the correct version of Python. We went to run the program and received the same error again. There was not much left to try with Pytbull so we moved on and went with a different approach to generate traffic.

10. ARMITAGE

Since we weren't able to generate traffic using Pytbull, we learned that we could use Armitage to send malicious attacks to our Snort Box. This program is a graphical interface that can be installed on Kali Linux and uses the Metasploit framework. Once Armitage started up, we set our Snort Box as the target and launched every attack it could produce; known as the Hail Mary. This tries everything in the Metsploit database and results in a lot of malicious activity. While it was running, it set off a lot of alerts and our intrusion detection system picked up everything. The log was spitting out attacks after attacks from the Kali Linux laptop running Armitage. We finally were able to create traffic for Snort and our test was successful.[1]



Attack 10.156.189.141

Wordpress InfusionSoft Upload Vulnerability

This module exploits an arbitrary PHP code upload in the WordPress Infusionsoft Gravity Forms plugin, versions from 1.5.3 to 1.5.10. The vulnerability allows for arbitrary file upload and remote code execution.

Option	Value
LHOST	10.156.189.157
LPORT	12594
Proxies	
RHOST ✚	10.156.189.141

Targets: 0 => Infusionsoft 1.5.3 - 1.5.10

☐ Use a reverse connection

☐ Show advanced options

```
[*] Scanned 1 of 1 hosts (100% complete)

[*] 1 scan to go...
msf auxiliary(ssh_version) > use scanner/mysql/m
msf auxiliary(mysql_version) > set THREADS 24
THREADS => 24
msf auxiliary(mysql_version) > set RPORT 3306
RPORT => 3306
msf auxiliary(mysql_version) > set RHOSTS 10.156.189.141
RHOSTS => 10.156.189.141
msf auxiliary(mysql_version) > run -j
[*] Auxiliary module running as background job
[*] 10.156.189.141:3306 is running MySQL, but responds with an error: \x04Host '10.156.189.157' is not allowed to connect to
[*] Scanned 1 of 1 hosts (100% complete)
```

Figure 5: Deploying an attack

11. ISSUES & DELAYS

Throughout the project, we encountered some issues and delays. First, it took some different documentation to piece together everything we needed for Snort to run correctly. Snorby was not able to sync with our alerts, but we could install the graphical interface. Once we had everything up and running, we needed traffic to test our intrusion detection system. We ran into issues when trying to start Pytbull, the IDS/IPS testing framework, because all of it's components haven't been updated in over two years. We tried two different virtual machines and neither them worked when running the python software Pytbull.

11.1 Solutions

It took two tries to install Snort because the first we were missing things for it to fully run. After gaining information from different sources, we were able to get a fully functional install of Snort. We even created our own guide for future references. Snorby was not playing nice with our project and we ended up leaving it out of the final presentation. We only wanted it for demonstration purposes to show others the organized graphical interface it produces. We also had to give up on trying to get Pytbull working because it seemed to give us python errors no matter what we tried. A fellow member of the NCS club suggested we try using the Hail Mary attack which was built into Armitage. We installed the necessary software on a Kali Linux Laptop and ran the attacks against our host. This method proved to be very successful as we saw alerts flooding our log non stop. Now we could finally tell that Snort was doing its job and we have created a working intrusion detection system all from open source.

Ever since Snort was bought out by Cisco there has been a decline in development of third party application for snort. Although Snort is still open-source and they provide a set of rules for everyone and registered users, it seems that Snort is now used internally and the learning curve has become more steep.

12. FUTURE WORK

With this project, we can look more into Snort and see that it is capable of running as an Intrusion Prevention System as well. If we have the necessary equipment and set up a router with our systems, we can use the full power of Snort. After looking through the logs of alerts our detection system found, we can go ahead and start blocking any malicious attacks from unwanted IP addresses. Instead of just seeing everything that is going on, we would be able to do something about it with more control over our network.

After installing Snort and trying out the different third-party tools we realized that we had to setup an environment which used older versions of ruby and python. We felt that a IDS should not be using outdated python and ruby libraries because they could contain security flaws. We focused on getting Snort itself working properly and making sure we could test its rules.

13. CONCLUSIONS

In our project, we were able to learn a great amount about Intrusion Detection Systems. Reading about them and memorizing how they work just doesn't teach as much about them as setting one up yourself. Completing a hands on project for this topic really allowed us to go through all the steps and documentation and try different methods until we finally got it working. We encountered a few errors along the way, but that's what learning is all about. When we came across these issues, it did slow our progress down, but we realized we must go out of our way to find alternative solutions to fix it. Snort was a great project for us to experiment with because it is open source software. We could try different things during the installation and had the power to make changes to the configuration. We could create our own rules based on alerts we wanted and even use the pre-installed ones. Overall, this project was a great experiment and we both learned a lot about intrusion detection systems and other related software and applications. Hopefully we come across these services and tools again in future work and can impress employers with our knowledge on how they operate.

14. REFERENCES

[1] Getting started with armitage and the metasploit framework (2013).
[2] Snort subscriber rule set categories.
[3] Overview of intrusion detection & prevention, 2011.
[4] Pytbull, October 2012.