INTRUSION DETECTION USING SNORT

R Siva Girish
Computer Science
PES University
Bengaluru, India
sivagirish81@gmail.com

Gaurav C.G
Computer Science
PES University
Bengaluru, India
officialgauravcg492@gmail.com

***Abstract*--SNORT is an open-source network intrusion detection system. Network intrusion detection systems are used to monitor network traffic, analyse incoming packets and determine whether the incoming traffic is healthy traffic for the network or simply a bogus request by an attacker. Network intrusions can be of various forms such as a denial of service, man in the middle, nmap scans. Snort when equipped with a robust ruleset can handle these attacks and can also be programmed to alert or drop malicious packets as the case may be. Our rule set consists of rules to detect man in the middle attacks, usage of reconnaissance tools like nmap and a few variations of denial of service attacks.**

Keywords--SNORT; Network intrusion detection system(NIDS); Rules; Packets; alerts; Time to live(TTL);

I. INTRODUCTION

Our Ruleset is catered towards detecting attacks in our network. Whenever malicious activity or packets are detected. Snort alerts the user stating that the incoming packets are malicious or that the incoming requests to the server are bogus. This gives the user the discretion to drop these packets/attackers from their network and deny them any further access.

We're using the SNORT NIDS to implement this, which enables to write our own rules to detect various attacks. Snort can function in three modes - :

- Packet Logger
- Packet Sniffer
- NIDS

For our purpose, we use Snort in NIDS mode and if required we can use it to sniff as well as log the incoming packets to a log file. This log file can later be used to analyse the incoming traffic and write rules accordingly. In our rule set, we have started by accounting for all the packets that are being transmitted in our network like ICMP, TCP and so on.

Further, we move on to detecting the usage of reconnaissance tools such as nmap which is of high importance as the first-ever measure taken by any hacker is to scan the various ports in a machine to determine the type and the nature of the device to be hacked.

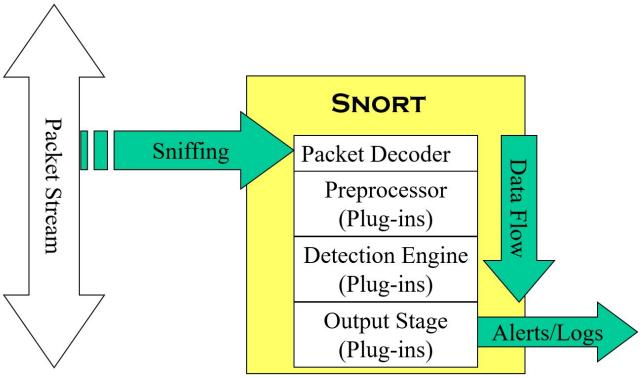
DOS attacks are very common and amazon themselves were affected severely by a massive DOS. Our ruleset will be able to detect a few variations of the dos attacks such as dos syn flood, fin flood and ping of death.

Man in the middle is a type of attack that makes the connection across two computers compromised as the phisher acts as the router and all communication between the two computers are now directed by the phisher which is a violation of the privacy of the two end hosts. Our ruleset has rules channelled towards detecting these types of attacks and alerting the user.

II. MOTIVATION

Network Security as a domain has developed massively over the past few years.Due to lot of cyber attacks executed by talented hackers on highly protected networks. In most cases data is stolen and misused.This is something that is strongly ridiculed and many organizations are actively working against securing their networks against hackers.
Snort as an intrusion detection system can be a huge challenge factor for hackers. Most hackers hack a network when the network is highly vulnerable or by reducing it to a vulnerable state .
Snort can act as an additional layer of security for an organization or to an individual system. The capability of snort to detect attacks depends entirely on the rule writing.. With unsupervised rules snort might just be an ordinary packet sniffer, so the performance of snort is dependent on the quality of rule written. Most snort rules are proprietary though the software is open source. Hence we wanted to contribute to the snort open source projects by writing few rules for attacks which we believe are still very popular. In this paper we discuss a few of the attacks for which rules have been written with their corresponding detection output.

III. ARCHITECTURE



Source : Martin Roesch

Snort architecture is designed in such a manner that the data is sniffed from the packet stream and fed to the snort engine.The components of snort engine are

- Packet Decoder: The packet which enter the system are first passed to snort packet decoder which calls necessary decoders based on the protocol on the packet.
- Preprocessor : These decoded Packets are then checks for their correctness and validity of each packet before being transferred to the detection engine.
- Detection Engine : It is a tree based structure which contains the rulesets and maps each packet with the corresponding rule it triggers. The rule set is divided inside the engine in hierarchical fashion as the portion of the rule matches the search goes deeper down the nodes. If it reached the end node, then it knows that it needs to trigger a rule.
- Output : This component is responsible for the output rules which are written. If a rule defines to alert when triggered then the output is passed to alert plug-in.

We deal with the detection engine where the incoming packets are compared with the rules written. We define these rules such that snort detection is made efficient..

IV. Problem Statement

Network intrusion detection in the modern world is of paramount importance. As the number of people connected to the internet is increasing every day the potential threats such as cybercrimes are also constantly on the rise. Therefore it is important to know whether the network a person is connected to is secure or not. The cheapest and most feasible solution to this problem is to have a NIDS that can detect malicious packets, requests and drop any further request from the user that is sending them.

V. Rule Set

Snort is an open source software but the rulesets to detect various attacks are not necessarily open source.There are a lot of rules contributed by the open source community but a fair amount of the are unable to serve their purpose to detect the attack they are supposed to detect.Our goal was to develop our own open source rule set to detect a few selected attacks.In the process we have managed to detect man in the middle attacks , a few types of dos and nmap scans.We have tested these rules by attacking a host using a VM running Kali Linux which is one of the most advanced open source penetration testing software.
Developed rules to detect malicious packets as well as irregular traffic.Based on the gravity of the attack the user can decide to drop those incoming requests and proceed to service other meaningful requests from legitimate users.

Hackers can hack a network in many different ways. Typically a hacker starts by scanning all the available ports in a server system, Depending on the nature of the response received from the system the hacker gains knowledge about as to whether the server is active and the type of machine that responds the request. This is the basis for running nmap scans. Similarly, we have developed rules for attacks that involve a denial of service. Due to heavy load on the server caused by hackers by flooding the server with syn or fin packets leading to the server wasting a lot of time servicing requests which come for the sole purpose of disrupting services to other legitimate users.

Lastly, the man in the middle attacks which involve monitoring activities between two hosts by spoofing their networks router and altering the communication channel between the two hosts to make sure all subsequent requests and responses between the two systems go through the hacker's system.

Based on the above facts we crafted rules for - :

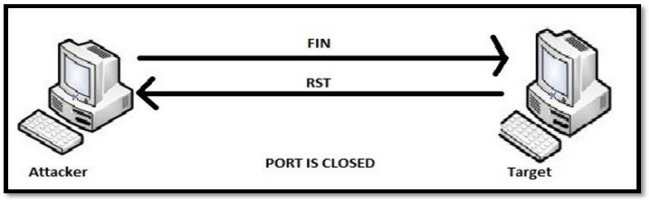
- Nmap Scans
- Denial Of Service
- Varied TTL
- Man in the middle

VI. Implementation

With the premise set to develop rules to defend networks against the aforementioned attacks, we have crafted rules for each of the attacks :

A. Nmap Scans

- Nmap ping Scan - Used for identifying the status of the receiving host by sending ICMP packets. Detected by monitoring all the ICMP packets incoming to the network.
- Nmap TCP Scan - Used for network enumeration like TCP handshake. Detected by monitoring all incoming TCP packets.
- Nmap Xmas Scan - Attackers use Xmas scan to initiate a connection with a server by sending FIN, PSH, URG packets rather than the traditional TCP handshake by sending SYN/ACK. Detected Nmap Xmas scan by setting the flag in our rules to detect FIN, PSH and URG packets coming together.



- Nmap Fin Scan - Attacker sends a fin packet and tries to connect with the server. Detected Nmap fin scans by setting the flag value to FIN and alerting all such packets.
- Nmap NULL Scan - Attacker may choose to communicate with the server or send data packets through none flags only. Detected Nmap NULL scans by setting the flag value to null and alerting all such packets.

B. Denial Of Service

A DOS attack is one of the well-known threats, which focuses on stalling the server from servicing legitimate requests and sometimes cause complete shut down of the system. Few of the attacks for which these rules are developed are listed below.

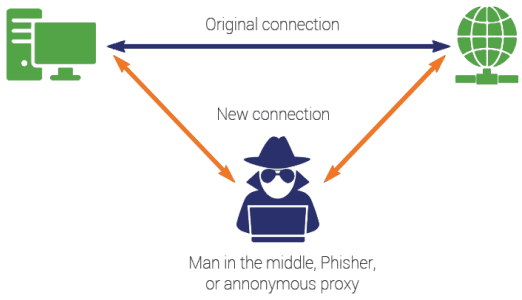
- SYN flood - disrupts the functioning of the server by continuously sending SYN requests to the server with a spoofed(fake) IP address. The server without this knowledge will repeatedly send SYN-ACK request to establish the connection and would deplete its resources. Due to this legitimate requests are unacknowledged. Detected by checking the SYN requests sent from a particular source, if a certain threshold is crossed then we alert such packets.
- FIN flood - similar to SYN flood but sends FIN packets which tell the server to close the connection. But the server will be confused because there is no such existing confusion. Many such requests cause the server to crash. Detected the same way as above but with the flag set to FIN.
- Ping of Death - it causes failure at the server by sending fragmented ICMP packets to the server. Normally the total size of a packet should be less than 65535 bytes, but by sending fragments which would total up to more than 65355bytes, the server would be overflowed when trying to combine these fragments which stops the functionality of the server. Detected by seeing the size of each fragment and the number of fragments received. If this count is higher we alert such packets.

C. *Varied TTL*

TTL is the maximum number of hops that can occur between two routers. With each hop, the TTL value decreases by one. The standard value of TTL for most purposes is 64. Whenever a packet arrives with a TTL value other than that of the standard value such as 64 then we alert the host machine.

D. *Man in the middle*

Man in the middle attacks involves a third party that positions itself between one end host and another thereby illegally spying or modifying the packets. The goal of this attack is to either eavesdrop or impersonate another end host with intent to gain confidential information about the compromised users. This type of attack poses an immense danger to the end-users and hence must be prevented at all costs.



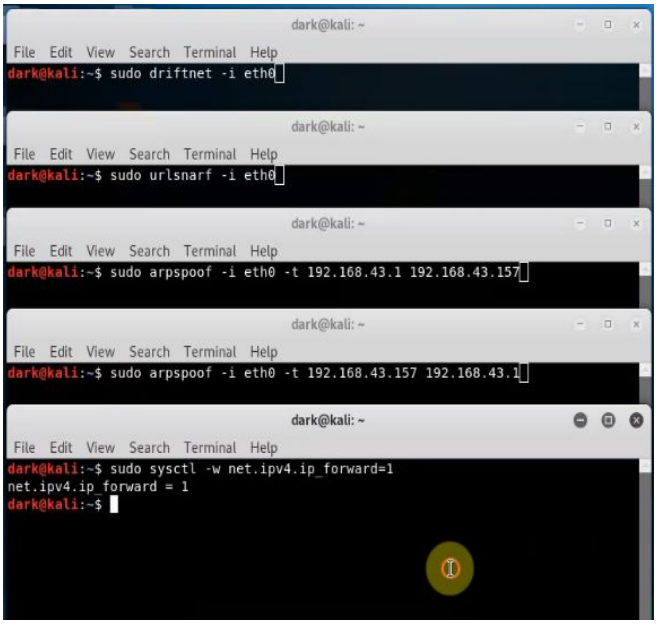
The approach we used to tackle this problem was based on the ICMP protocol. The ICMP Protocol is designed in such a way that it has the capability to detect the shortest pathway in the network between two end hosts.
Since in all cases of a man in the middle, the alternate host is introduced by the attacker and all the packets are being redirected to another host machine. This indicates that a longer route is being used despite the fact that a shorter route is available. Based on this concept we used Snort to detect ICMP packets of code 1 under type 5 which indicates that the datagrams have been redirected despite a shorter route being available.

VII. Detailed Description

A. *Man in the middle*

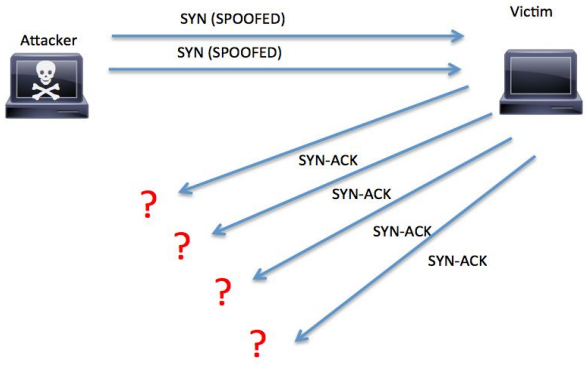
Here's an overview of how the snort engine is used to detect a man in the middle attack -:

1. First step in a man in the middle attack is to allow packet forwarding.As Packet forwarding is necessary for the hacker machine to act as a router.
2. Once Packet forwarding is enabled then the problem is reduced to arp spoofing. Arp Spoofing is a method by means of which a packets can be intercepted on a switched LAN.
3. We arp spoof to start intercepting packets from the victim to the router as well as the router to the victim i.e technically at this point all communications between the two end machines will be directed through the attackers machine.
4. Now that the attacker has launched a man in the middle attack on the host machine snort will be able to detect it.

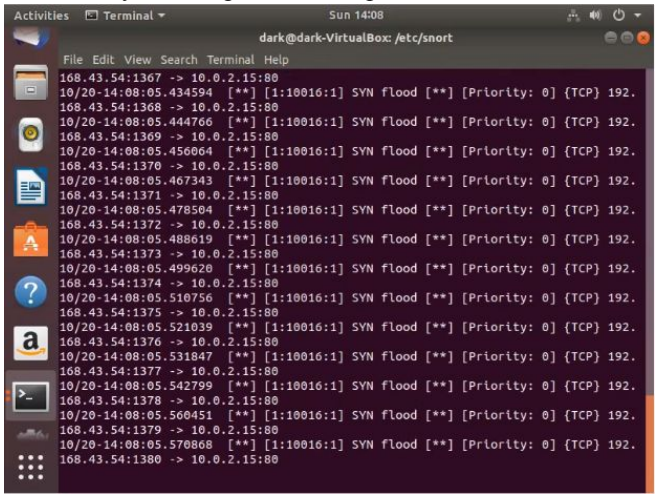


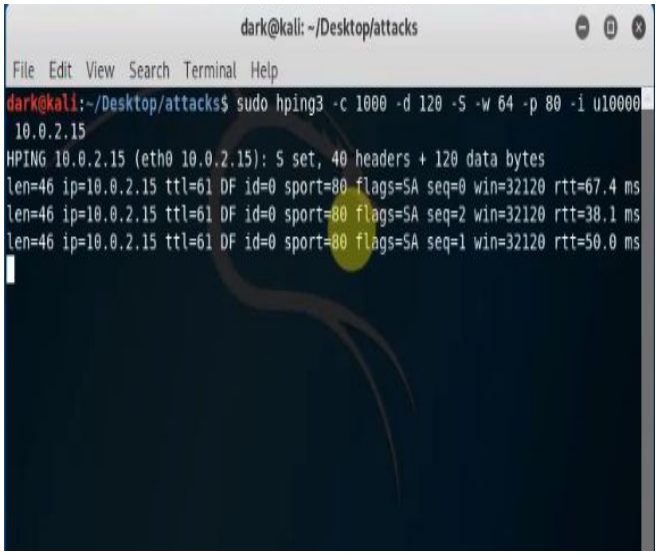
5. Snort running on the host machine will be monitoring for violations in any of its rules in the ruleset.
6. So as and when an icmp packet involves an unnecessary redirection (which is the case in a man in the middle attack) Snort will immediately alert the host machine regarding the intrusion.
7. Snort will alert the user stating that a man in the middle attack has been launched and snort can be configured appropriately to drop these packets or log them for future reference.
8. This is the general workflow of snort in cases of any rule violation.

B . Denial Of Service



The first attack discussed is SYN flood. This attack is first setup using an attack VM like Kali Linux, hping is the tool which is used to produce 'n' number of SYN packets within for every time 't'. By setting the value of 't' in milli or nano seconds we can produce large number of packets per second. The tool also takes the IP address of the defender machine as an argument. The defender machine is then set to receive a flood of SYN packets. This attack is detected in snort using a feature of snort known as 'detection_filter' which has additional properties like 'track', 'count' and 'seconds'. By tuning these parameters one can detect whether those packers arrive at the given rate. The flag option in the ruleset is set to 'S' indicating SYN packet. Since this is a vulnerability in TCP protocol, the protocol ID is set to TCP.





Similarly FIN and PSH/ACK floods can be performed using hping with flags set to 'F' and 'P/A'. They can be detected with similar rule which was defined for SYN but with change in the corresponding flag bit.

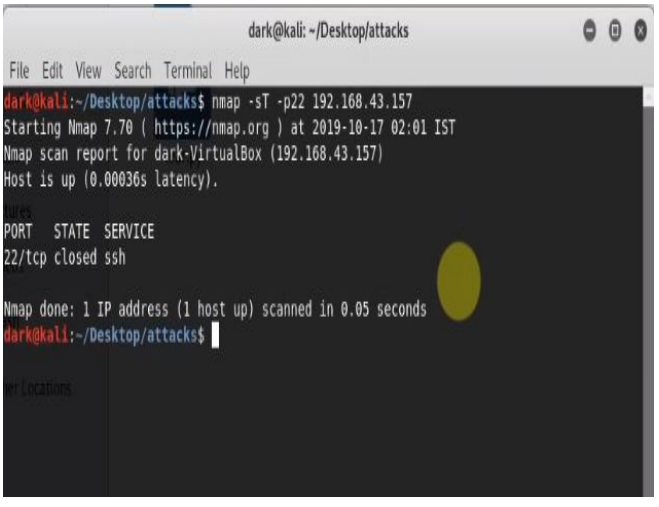
Ping of Death is performed using 'hping' or by a python module named 'scapy'. We send many ICMP fragments each sized around 1000 bytes which in total would be greater than 65355 bytes which violates the ICMP protocol. If a system receives such fragments it is bound to crash trying to unfragment it. This is detected in snort by setting the 'dsize' option to be greater than 1000 and using 'detection_filter' option to alert is the number of such packets total to a sum greater than 1000.

All the above attacks discussed are transport layer associated.

SlowLoris is an application layer attack which targets the http requests. Usually in a client server architecture, the client would first establish a TCP connection and then would send an HTTP request. But in slowloris, we send incomplete HTTP requests were the request headers are passed with an interval. In this way the server is bound to wait for the complete request to service it. This attack is performed by using scapy module of python and this was detected using 'content' option where the value passed to it is a regex of junk header request which is a general format. Then 'detection_filter' is used to check the frequency of these headers send. SlowLoris is names as slow because it sends incomplete headers at a slow rate. Hence the rate defined by us should also be less.

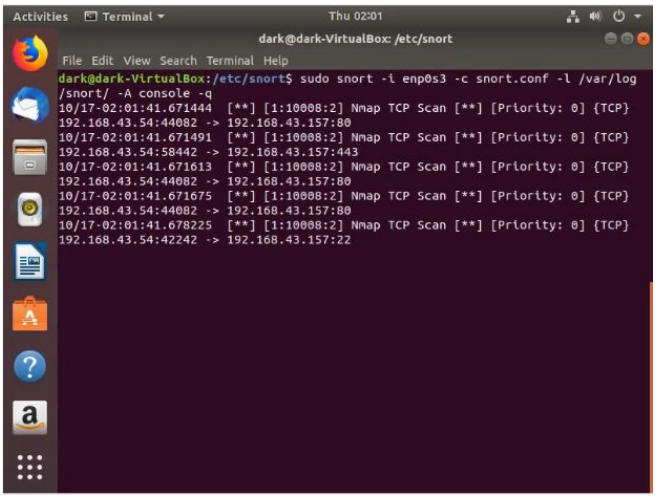
C . Nmap Scans

Nmap scans in general involve transmitting different types of packets across two end systems therefore it is sufficient to detect the type of packets being sent between two systems.This logic works in most cases but in cases where an nmap ping scan or a TCP scan is involved the rules written become noisy as any query for a website involves TCP packets and any transfer of icmp packets will trigger the rules unnecessarily. However in cases where a null packet or a fin packet is being transmitted in case of null scan or fin scan these rules are less noisy and detect accurately and are hence extremely useful.



1. Ran various types of nmap scans like ping,TCP,null,fin and xmas.

2. Used Snort to monitor the incoming packets.
3. Depending on the type of packets detected.



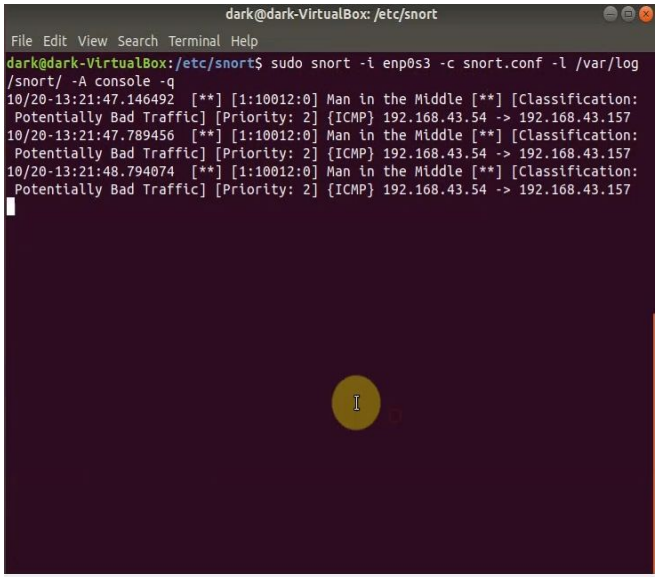
4. Mapped the incoming packet data to an appropriate nmap scan.

D . Varied TTL

While detecting packets which have an abnormal value we use scapy tool which is an open source packet generator to generate packets with a ttl value other that 64.Used snort to monitor the ttl value of the incoming packet and alert the host machine whenever a different ttl value is observed.

VII. User Interface (UI)

The User Interface is restricted to the terminal or the shell in which the snort tool is run. The snort is usually run in daemon mode and alerts are written to a file which can be specified prior to starting the application. For convenience we run snort in terminal as a standalone application and we specify a value '-q' which indicates quiet and does not display anything when it is running. All the triggered rules will automatically be alerted on the terminal itself to the user and it would be up to the user to drop or allow these packets or users as the case may be.



The above diagram tells the user that he is being arp spoofed and that he is being monitored by means of a man in the middle attack. In the above picture, snort is running in quiet mode and the default interface in which snort is running is in enp0s3.

IX. Outcomes

Snort has a fully active community supporting it's open source initiative and we have made a small contribution to the open source community by making our rules available to the entire community.

Most of the rules designed in the open source community rulesets are not perfect and often fail to fulfill the purpose they were created for.Hence the need to develop new rules and to perfect the existing community ruleset.

With our goal set in our mind we took up some common attacks that are executed by hackers on other host machines. Based on which we were able to learn how to attack as well as detect the rules on snort by using two virtual machines, one running Kali Linux(Attacker) and the other running Seed Ubuntu the defender machine running Snort.

With this setup we were successfully able to attack the seed ubuntu machine as well as defend it by running snort.

This has broadly improved our understanding of networks and security aspect of it. We were able to model the real time scenarios using these two virtual machines.

We also understood some of the flaws in the current TCP/IP protocol and how they are being exploited for ulterior motive. We were able to map our learning to actual real life intrusions and were able to detect them as well.Giving us in depth understanding of some of the attacks and their manner of execution.We were able to translate this understanding to a rule to detect an attack and defend the end system against that specific attack.

X. Acknowledgements

[1] Prof. Honnavalli B Prasad, our guide and mentor for this project.

[2]ISFCR PESU, providing us with the resources to test our Project.

XI. References

[1] https://www.snort.org/

[2] Snort2.1 Intrusion Detection by Jay and Caswell

[3] Advanced Intrusion Detection using Snort, Apache, MySql, PHP and ACID

[4] Snort IDS and IPS toolkit