CONTENTS

1. Intrusion Detection and Intrusion Prevention Systems (IDS / IPS)

Intrusion detection systems are one of the important security components, which include methods that enable monitoring of activity on the network and analysis of traffic to detect possible attacks, violations and threats. Intrusion prevention systems are network security systems that cover the detection and prevention of attacks. Today, networks have a complex structure, they are connected to other networks with many access points, especially the internet, cyber attacks are diversifying and increasing day by day, and at the same time, these complex network systems can no longer be protected only with encryption or firewall. It has made it inevitable to detect the attack attempts in real time.

IDS / IPS systems have functions such as frequently monitoring the network, identifying potential threats and keeping event records (logs) related to them, stopping attacks and reporting to security administrators. In some cases, these systems can be used to reveal weaknesses in the security policies of the institutions. IDS / IPS can also detect attackers' network-related information gathering activities, and they can stop the attackers at this early stage.

2. What is The Snort?

Snort is a network intrusion detection system developed in 1998 by Martin Roesch. Snort, an open source and free software distributed under GNU license, is currently developed by Sourcefire, a company founded by Martin Roesch. The software can be run on various operating systems such as Linux distributions, Windows and MAC. There are also independent company software developed to operate with graphical interface support based on Snort software. These softwares perform functions such as management, reporting and logging.

3. Features of Snort

- Real-time traffic monitor
- Packet logging
- Analysis of protocol
- Content matching
- OS fingerprinting
- Can be installed in any network environment.

- Creates logs
- Open Sourse
- Rules are easy to implement

4. Purpose of Snort

The main purpose of Snort is to perform packet logging and traffic analysis on the network. But in default settings, Snort can handle a maximum of 40-50 mb of traffic. In order to increase this admission traffic, the default settings can be changed and the traffic bandwidth can be increased. Thanks to its flexible architecture, Snort can detect many attacks and malicious / suspicious software. Since Snort is a rule-based IPS system, it can control both signature and other components. Another convenience of Snort compared to other IDS / IPS systems is that it allows you to write flexible rules. Snort rules can be written manually or ready-made rules can be used.

Snort basically can look at the data in all layers up to the application level and collect certain traffic from this data. It then evaluates what it finds by applying the rule sets that can be defined by the user or the developer. However, in some security scenarios, looking at only one package may not be sufficient to detect the attack. For example, the attacker may have sent packets to be fragmented so that they will not be detected, or the attacker may have performed a port scan. In such cases, these fragmented packages need to be combined. Preprocessor module in Snort architecture can do this merging process. Another situation where preprocessors are used is the need to decode the encoded data.

5. Snort Platforms

Snort can run on many operating systems and hardware. The following are some of them:

- Linux
- OpenBSD
- FreeBSD
- NetBSD
- Solaris
- HP-UX
- AIX
- IRIX

- MacOS
- Windows

6. Attachments Used With Snort

BARNYARD: It allows the records of Snort while listening to the network to be written into the database within a certain template.

ADODB: Provides the connection between the database and BASE.

BASE: Provides the snort records that Barnyard writes to the database to be presented to the user with more understandable graphical tables through a web interface.

OINKMASTER: It is a tool designed to regulate the snort rules, making it easier for the users.

PULLEDPORK: It is a tool that makes it easier for the users to update the snort rules in our system when the snort rules are updated.

SNORTSAM: It is a snort extension that provides automatic IP blocking in the firewall.

ACID: It is a PHP-based analysis tool that enables the data written to the database by intrusion detection systems to be retrieved and processed and presented to the user on a web interface.

SNORBY: It is a web application developed with ruby on rails that provides security status monitoring. It can work with popular IDSs.

7. Advantages and Disadvantages of Snort

Snort monitors the network and the computer for any attacks free of charge and provides an error when necessary to inform and protect us against these attacks. Thusly, Snort is the best alternative to the more expensive commercial IDS systems because it has important benefits that it provides us. In this case, there are some advantages and disadvantages of using snort. These can be listed as follows:

7.1. Advantages

- Snort provides open source and free monitoring for network and computer.
- Snort does not carry licensing costs or software maintenance updates. You can deploy Snort in an organization filled to the gills with money or one that has no budget to speak of.
- It effectively prevents any damage to the network.
- Snort is quickly installed and running on our network or computer.
- Snort has good support available on the Snort website.
- Any alterations to files and directories on the system can be easily detected and reported.
- It provides a user-friendly interface which allows easy security management systems.
- Snort is cross-platform. Thusly, It can be installed on Windows NT, Windows 2000, HP-UX, Solaris, OpenBSD, FreeBSD, NetBSD, Linux, MacOS X, and many more UNIX flavors and processor architectures.
- Snort does not need to supplant any existing security infrastructure. Rather, it complements existing commercial products quite nicely.
- There is an active community of users and developers. Alert signatures are often available immediately, if not within hours, of documented attack behavior. Bug reports and feature requests often are addressed directly by the development team, who participate in the snort-devel and snort-users mailing lists. There are many add-on utilities that are not part of Snort proper, but offer additional features and ease of use.
- Properly configured, it gives a good overview of what's going on in the network, and provides a way of automatically logging packets from potential attacks for future reference. With some careful thinking, it can even be used for reacting directly to attacks as they occur.

7.2. Disadvantages

- When deploying Snort, it's important to make sure the used rules are relevant and up to date, otherwise the system will be much less efficient due to low signal-tonoise ratio in the case of a bad choice of rules, and due to Snort missing attacks completely in the case of a Snort system with rules not being updated properly.

- Apart from the challenge of choosing/writing good rules for Snort, there is also a related disadvantage. SinceSnort only looks for things defined in its ruleset, it doesn't have the ability to tell what traffic is considered to be normal from each host on the network, and what traffic seems to be out of place. This way, 'normal' behaviour but from the 'wrong' computer on the network can not noticed unless rules are setup on a host-by-host basis.
- The administrator must develop customized logging and reporting methods.
- Snort does not support token ring.
- Although Snort is flexible, it does lack some features found in commercial intrusion detection systems.
- Snort rules must be developed carefully. This is necessary to reduce the number of false alarms of information generated and to reduce the amount of information logged.

8. Snort Architectural Structure

Snort is an open source network-based IDS. It is the most preferred intrusion detection system with more than 4,000,000 downloads and approximately 500,000 registered users. It can perform real time network traffic analysis and packet logging. It detects attacks by analyzing protocol and content on packets it listens from the network. It can detect buffer overflow, port scanning, CGI attacks. Snort basically can look at the data in all layers up to the application level and collect certain traffic from this data. It then evaluates what it finds by applying the rule sets that can be defined by the user or the developer. However, in some security scenarios, looking at only one package may not be sufficient to detect the attack. For example, the attacker may have sent packets to be fragmented so that they will not be detected, or the attacker may have performed a port scan. Thusly, these split packs need to be combined. In this case, components in the Snort architecture are used. Preprocessor module in Snort architecture can do this merging process. Another situation where preprocessors are used is the need to decode the encoded data. For example, consider a scenario and the snort will alert when it detects the word "exploit". But the attacker divides the word exploit into 2 packages and sends it as "expl" and "oit". In this case, if these packets are not combined, the attack will not be detected. For this reason, with any preprocessor module in Snort, it first combines the fragmented packets and then passes it to the next stage for intrusion detection analysis. In another scenario, when the Snort system searches for ASCII content, the attacker sends the packet content as URL encoded. Due to the encoding process, the snort system

cannot detect the attack. However, as a result of decoding the data coming from the attacker by preprocessor modules, the attack can be detected by Snort. As a result, Snort is made up of different components, and these components work together to identify attacks and generate output. Snort-based IDS systems mainly consist of the following components:

- **Packet Decoder:**

It receives various types of packets from the network interface and prepares it for the next snort component.

- **Preprocessors:**

Before the Detection Engine prepares the packets for it and tries to determine the attack according to the anomalies in the packet headers. Preprocessor often works to piece together large chunks of data that come in bit by bit.

- **Detection Engine:**

It is one of the most important components of snort. It is responsible for determining the types of attacks within the package. It also uses snort rules to detect these attacks. An attack signal specified in the snort rules is compared with the packet content from the preprocessor component, and if there is a dangerous situation, the next module is passed to perform the action in the rule header.

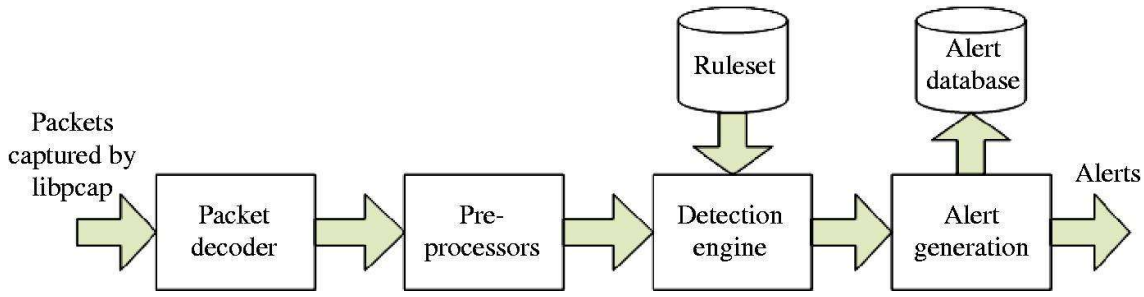
- **Logging and Alerting System:**

Detection engine component performs operations such as logging and warning according to the action specified in the rule header after determining which rule the package content corresponds with. Logs can be saved to a simple text file in tcpdump or any other file format. In the command given for the snort to work, the directory to be logged can also be specified with the -l parameter.

- **Output Modules:**

Allows you to route or use the output of the Detection engine component in other ways. Often outputs are initially saved in the / var / log / snort directory. The output obtained using this module can also be sent to another destination. The output produced by the Output or 'Logging and alerting system' components can be used in different ways. Snort logs can be kept systematically in the database. For example, we can keep the snort

outputs in a MySQL database, send it to Windows as an SMB pop-up, and send it to both SMB Windows and save it in the database.



9. Usage Of Snort

9.1 Cyber Security Solutions Provided by Snort

Snort is a very popular open source network intrusion detection system (IDS). In this case, it has cyber security solutions provided to us. First, the main purpose of Snort is to do packet logging and traffic analysis on the network. Thanks to its flexible architecture, Snort can detect many attacks and malicious / suspicious software. Since Snort is a rule-based IPS system, it can control both signature and other components. In addition, Snort offers us the solution to write flexible rules. Snort rules can be written manually or ready-made rules can be used. Also, Snort can also be used to perform protocol analysis, content searching and matching. And it can be used to detect various types of attacks such as port scans, buffer overflows, etc. It is available on Windows, Linux, various UNIX as well as all major BSD operating systems. Although the Linux version has a wider following and offers many feature-enriching add-on tools, support for the Windows version is growing. The latest version for Windows, Snort 2.0, supports most of the functions that the latest version for Linux, Snort 2.0.0, offers. The best cyber security solution provided by Snort is that Snort doesn't require that you recompile your kernel or add any software or hardware to your existing distribution, but it does require that you have root privileges. Thus, an additional software and hardware expense for cyber security is avoided. It's also possible to have Snort interact with other services - for example, it's possible to use it with iptables or similar firewall software for auto-blocking hosts that seem to be doing things they're not supposed to do. In this case, apart from simply logging the packets, Snort can for instance be told to take out the IP address of the potential attacking host and pass it on to the firewall software, telling it to block the host. We can configure the snort in 3 different modes and use it.. In the sniffer mode of operation, Snort will read network packets and just display them on the console. In

packet logger mode, it will be able to log the packets to the disk. Also, Snort provides a real-time alert to the user. As expected, once an attack or an abnormal situation is identified, an alert can be triggered to the systems administrator to take corrective actions, as needed. In network intrusion detection mode, Snort will monitor the network traffic and analysis it based on the rules defined by the user. It will then be able to take a specific action based on the outcome. As a result, all it does is examine network traffic against a set of rules, and then alerts the systems administrators about suspicious network activity so that they may take appropriate action.

9.2. Reporting and Alerting

We can use the Snort configuration file, labeled snort.conf by default, to configure each sensor to forward data to your reporting or alerting service. This configuration file defines Snort variables, preprocessors, output plugins, and the locations and names of the rules files. Snort variables are used in rule processing to define network objects such as our home network or the location of our Web or DNS servers. The preprocessors provide additional built-in functionality to Snort and often store auxiliary or experimental features. Also, the output plugins define how and where the Snort sensor should send alerts and logs. Snort supports sending output in Syslog, tcpdump, MySQL, PostgreSQL, Microsoft SQL Server, XML, and SNMP formats, as well as a proprietary binary format. Last, the configuration file contains the location and names of the rules that you want to enable on that sensor. After a default installation and without any specific output plugins enabled, Snort logs all alerts to a file named Alert in the default log directory /var/log/snort. Snort appends new alerts to a file named alert and also logs each alert independently to a directory named for the alert originator's source IP address. The logs provide quick, file-level access to the alerts on each sensor. However, if we want to aggregate alerts across multiple sensors, we consider the database output plugin or syslog output plugin.

9.3. Snort Alerts

Alerts are placed in the Alert file in the logging directory (/var/log/snort by default, or the directory you specify with the -l switch). Snort exits with an error if the configuration file or its logging directory does not exist. Snort has 6 alert modes. Fore example, after the installation, we can specify what type of alert we want by supplying the -A command-line switch. We need to use the -A switch if we do not use the -c switch, or Snort will not enter

alert mode. Additionally, if we want to use just the -c switch, Snort will use the full alert mode by default. Snort alerts modes is as follows:

Fast:	In this mode, Snort will report the timestamp, alert message, IP source address and port and destination IP address and port. **(-A fast)**
Full:	Additionally to the fast mode alert, the full mode includes: TTL, IP packet and IP header length, service, ICMP type and sequence number. (**-A full**)
Console:	It prints fast alerts in the console. (**-A console**)
Cmg:	This format was developed by Snort for testing purposes, it prints a full alert on the console without saving reports on logs. (**-A cmg**)
Unsock:	It export report to other programs through Unix Socket. (**-A unsock**)
None:	Snort won't generate alerts. (**-A none**)

The mode Snort is run in depends on which flags are used with the Snort command. For example, to view IP packet headers at console, we can use this command snort -v . These flags available with the Snort command are listed as below.

Flag	Function
-v	View packet headers at the console.
-d	View application data with IP headers.
-D	Run Snort as a daemon.
-e	Show data-link layer headers.
-l	Run in packet logger mode.
-h	Log information relative to the home network.
-b	Log information to a single binary file in the logging directory.
-r	Read packets contained in a log file.
N	Disable packet logging.
-c	Specifies which file will be used to provide a ruleset for intrusion detection.

9.4. Snort Update

It's not enough to have Snort in use on the network and leave it at that when the installation is done. Like with any system, the administrator needs to make sure that the NIDS and its rules are relevant and up to date. A new exploit might spread quickly, and without corresponding rules Snort won't even notice it. Sourcefire, the company developing Snort, are constantly adding, deleting and modifying rules. It's possible to subscribe to updates to Snort rules - it's free for private use but costs quite some money for commercial use.

10. Snort Rules Structure

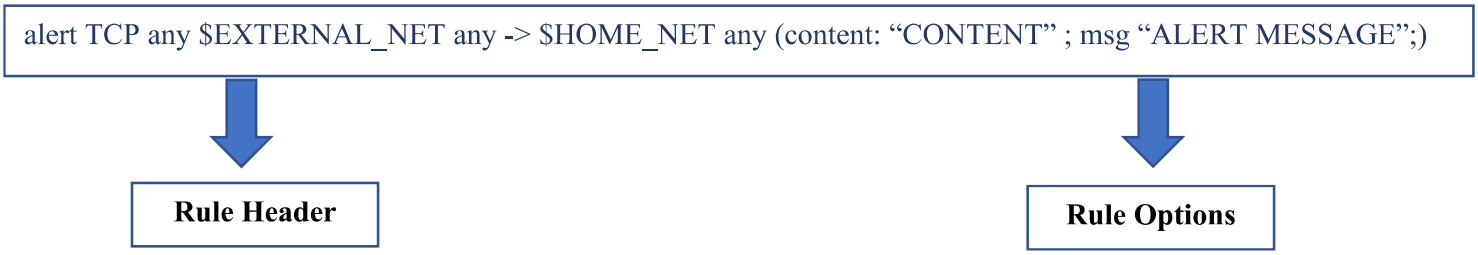
The real strength of Snort lies in its ability to employ rulesets to monitor network traffic. After successfully installing Snort, Snort examines outbound packets coming to the system based on predetermined rules and generates alarms and logs accordingly. In addition to the correctness of these alarms generated by the Snort, there are also alarms given incorrectly. Real attacks can be overlooked in this way and should be very careful when writing snort rules to avoid such a situation. The rulesets for Snort are contained within the lib files in the /etc/snort directory. The current release of Snort comes with 18 existing rulesets. Until you have gotten some experience using Snort and, more importantly, experience with the TCP/IP protocol suite, the best option is to use prepackaged rulesets, which are available from the Snort Web site. These rules are reliable and are available for almost any situation you might face as an administrator. Let's learn about the general structure of snort rules by examining an example snort rule. One can set an alert in the rules files with the following structure:

```
<Rule Actions> <Protocol> <Source IP Address> <Source Port> <Direction Operator>
<Destination IP Address> <Destination port > (rule options)
```

Structure	Example
Rule action	alert
Protocol	FTP
Source IP address	any
Source port	any
Direction operator	->
Destination IP address	any

Destination port	any
Rule options	(msg:"FTP Packet"; sid:477; rev:3;)

As an example, we can examine a rule like the one below. We will implement the real example of this after installation.



It is important to understand here that a few lines are added for each alert and these lines carry the following information:

- IP address of the source
- IP address of the destination
- Packet type and useful header information

Snort rules consist of two parts, the rule title and the rule option. The rule header contains the rule action, rule protocol, destination and source ip addresses, and destination and source port information. The other part includes the fields in which the specified action will take place and the properties related to the alarm message are defined. The part up to the parenthesis sign creates the rule header and is the part that should be included in every rule. The part within the parentheses constitutes the rule options.

10.1. Rule Header

Rule Action Rule action is the part at the beginning of the rule that determines what to do when the specified situation is encountered.There are some types of actions that may be performed by a Snort rule as follows.

Alert	Generates an alert message
Log	Logs the specified IP packet
Pass	Ignores the specified IP packet
Activate	Sends an alert message and then activates a dynamic rule
Dynamic	Activated by an activate rule, this rule then acts as a log rule
Reject	Paketi engeller, log kaydı alır, TCP RST ya da ICMP port unreachable mesajı gönderir.
Sdrop	Paketi düşürür ve log kaydı almaz.
Drop	Paketi düşürür ve log olarak kaydeder.

Also, the rule header defines the following properties:

- The action taken by the rule when an intrusion attempt is detected.
- The protocol the rule applies to.
- The source and destination IP addresses and netmask(s) affected by the rule.
- The source and destination ports.

Protocols

It is used to determine which protocol packets will be examined, located in the second place in the rule header. There are currently four protocols with which Snort can currently detect an attack. These are TCP, UDP, ICMP and IP protocols. Also, protocols such as ARP, IGRP, GRE, OSPF, RIP, IPX can be used.

IP Addresses

In the example rule, * $EXTERNAL_NET * and * $HOME_NET * are the parts that write their IP addresses. Also, the '!' Operator used to exclude can be used in these sections.

Port Addresses

Port addresses are used specifically to listen to a particular route. The parts reserved for the port numbers are the sections we wrote as "any" in the rule examples we have written before.

We can also create more specific rules by entering port numbers such as 80, 21, 22 in these sections.

Direction Operator

It is used to determine the transmission direction of packets to be examined. We use it to determine whether we will examine the packages that come out of our system or the packages sent to our system. There are three different direction determination methods. These are '->', '<- ',' < > '.

10.2. Rule Options

Rule options give flexibility and power to Snort rules, and they are the most important part of these rules. While creating the rule options part, a ";" character is put after each typed parameter. The rule options section defines these properties:

- Which part of an IP packet to inspect for compliance with a specific rule
- Any alert messages issued when an intrusion attempt is detected

The main rule options generally used in snort rules consist of the following:

Option	Function
Content	Searches incoming or outgoing packets for specified content.
Flags	Tests TCP flags on IP packets for specified settings.
Ttl	Checks the time-to-live on IP packets.
Itype	Looks for matches on the ICMP type field.
Icode	Looks for matches on the ICMP code field.
Minfrag	Sets the minimum value for IP fragmentation.
Id	Checks the IP header for a specified value.
Ack	Checks for a specified acknowledgement number.
Seq	Checks for a specified TCP sequence number.
Logto	Logs packets matching this rule to the specified file.
Dsize	Checks the size of the packet payload.
Offset	Used to modify the content option. Specifies the number of bytes from the start position to begin searching content.

depth	Used to modify the content option. Specifies the number of bytes from the start position to search.
msg	Specifies the message sent when a rule is matched.

The rule options given above are some of the most commonly used rule options. While creating snort rules, rule options vary according to the rule to be written, so different options may be needed. These can be easily searched and found on the internet.

10.3. Active & Dynamic Rules

Active and Dynamic rules are one of the strengths of the snort. Any rule that works with these keywords can activate another rule. An active rule using the 'activate' action header behaves the same as an alarm rule, only when defining it must have an 'activates' parameter in the rule option section. Dynamic rule uses 'dynamic' action header and behaves just like logging rules. While defining the rule option part 'activated_by' parameter must be used.

10.4. Writing Rules for Snort

It is very easy to create snort rules and protect the system with these rules. However, in order to create professional level snort rules and manage your intrusion detection system, it is also necessary to have an advanced understanding of network communication, architecture and protocols such as TCP / IP, OSI model. While specifying the protocol, it should be paid attention to have TCP, UDP, ICMP, IP protocols. While giving the IP address, it can be CIDR, netmask, "!" may be excluded by including. Likewise, "!" character can be excluded, the range can be given as "34: 9000", it can be specified as any port by using "any", ": 567" can be used in the form of ports smaller than 567. The direction of the traffic is indicated with the sign "->". We learn from this that the art of writing a good rule isn’t as easy as it might seem at first. The person deciding what rules to use and/or write needs to think about what should be considered normal traffic on his/her network, try to filter out as many false positive cases as possible.

11. Snort Usage and Applications

11.1 Snort Setup

Now let's move on to the snort setup. In normal operating systems, Snort should not run with root rights. Because Snort can be exposed to memory overflow attacks because it receives and processes inputs and packets coming from software and network. To avoid such a situation, we run Snort at normal user level in virtual machine. In the installation to be done on the Ubuntu 17.04 desktop version, we first made machine updates and then went to the installation phase. Below we have explained the setup of the snort step by step.

First, we update Linux ubuntu so that the snort can be installed properly. We use the apt-get update command for this.

```
apt-get update
```

We download the files to our server with the link below.

```
wget https://www.snort.org/downloads/snort/daq-2.0.7.tar.gz
```

After the files are downloaded, we open the compressed files on our server with the command below.

```
Tar xvzf daq-2.0.7.tar.gz
```

With the command below, we enter the compressed files in the directory where we unpacked them.

```
Cd daq-2.0.7
```

And complete the installation of the files we have opened with the command below.

```
.configure && make && sudo make install
```

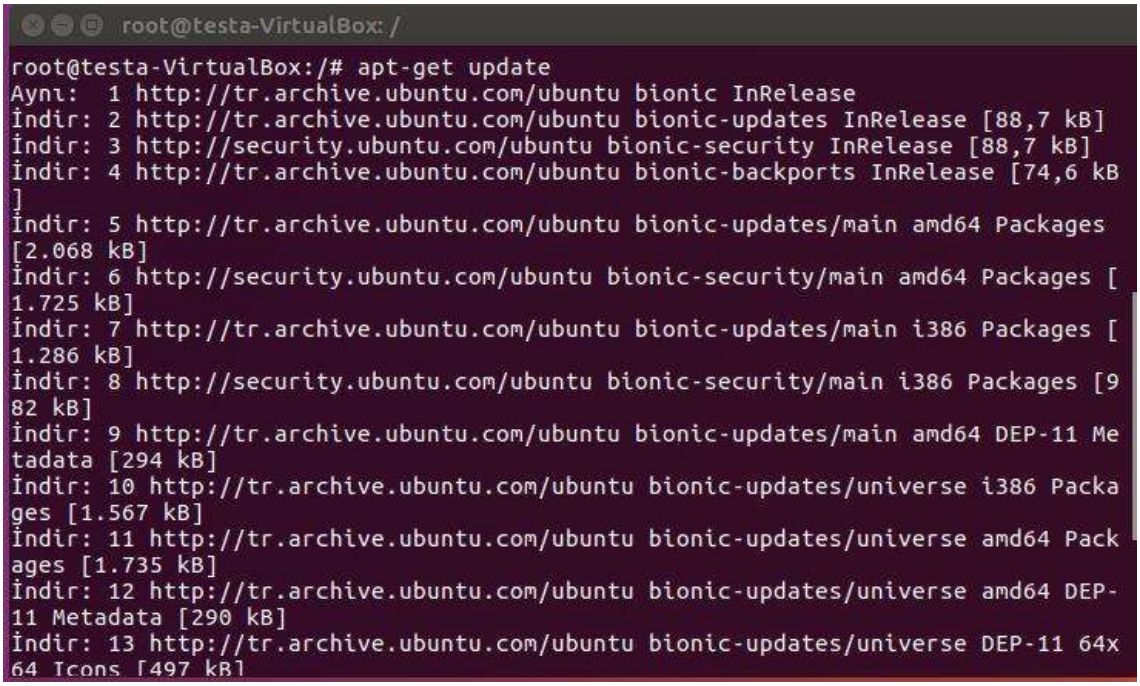
After "daq" installation is completed, we continue with Snort installation. For Snort installation, we will install 2.9.17.1 version with the following command.

```
wget https://www.snort.org/downloads/snort/snort-2.9.17.1.tar.gz
```

We decompress the files and enter the current directory and complete the installation of the extracted files as follows.

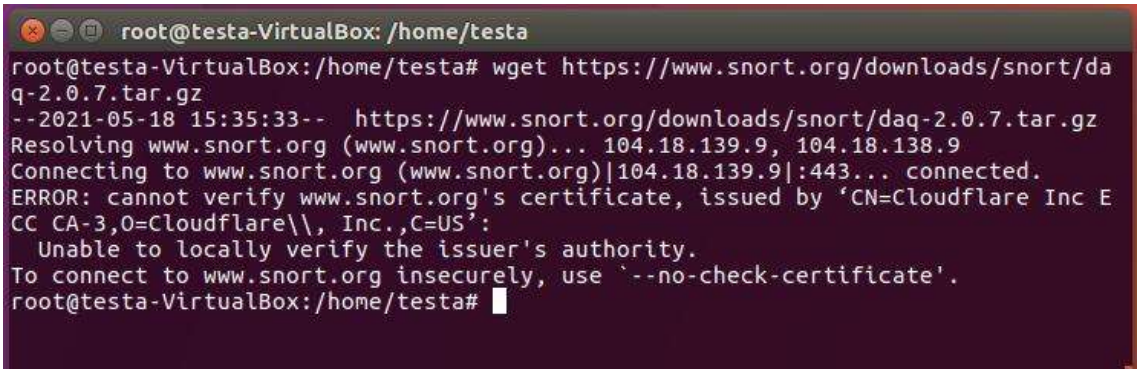
```
tar xvzf snort-2.9.17.1.tar.gz
```

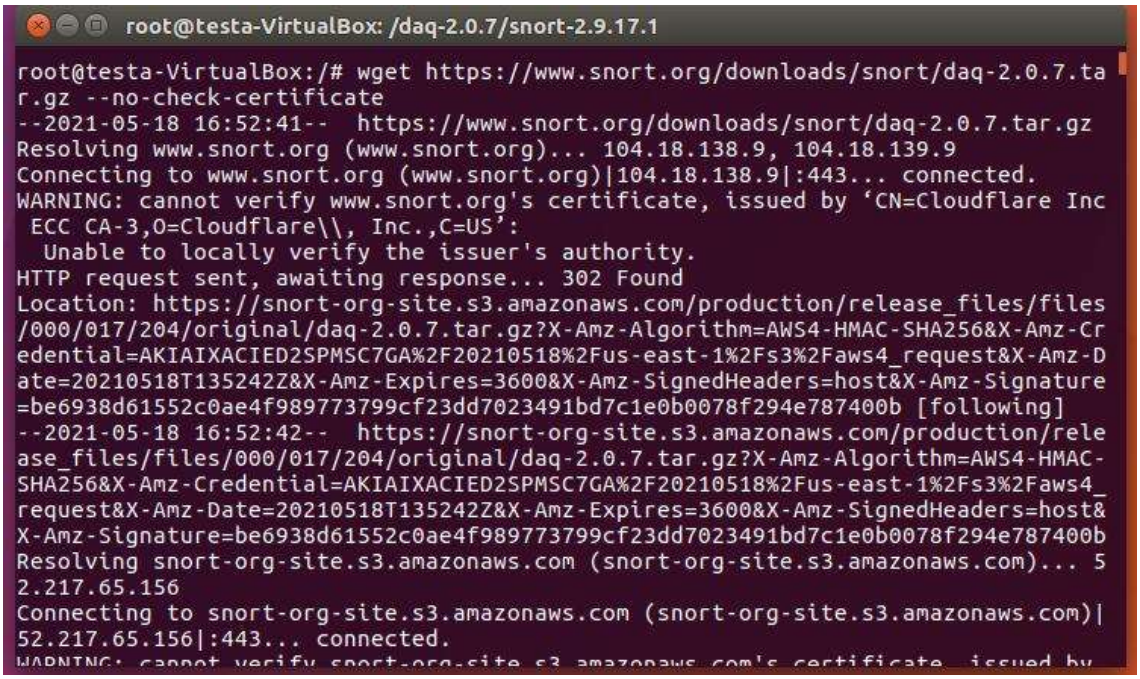
```
wcd snort-2.9.17.1

./configure --enable-sourcefire && make && sudo make install
```

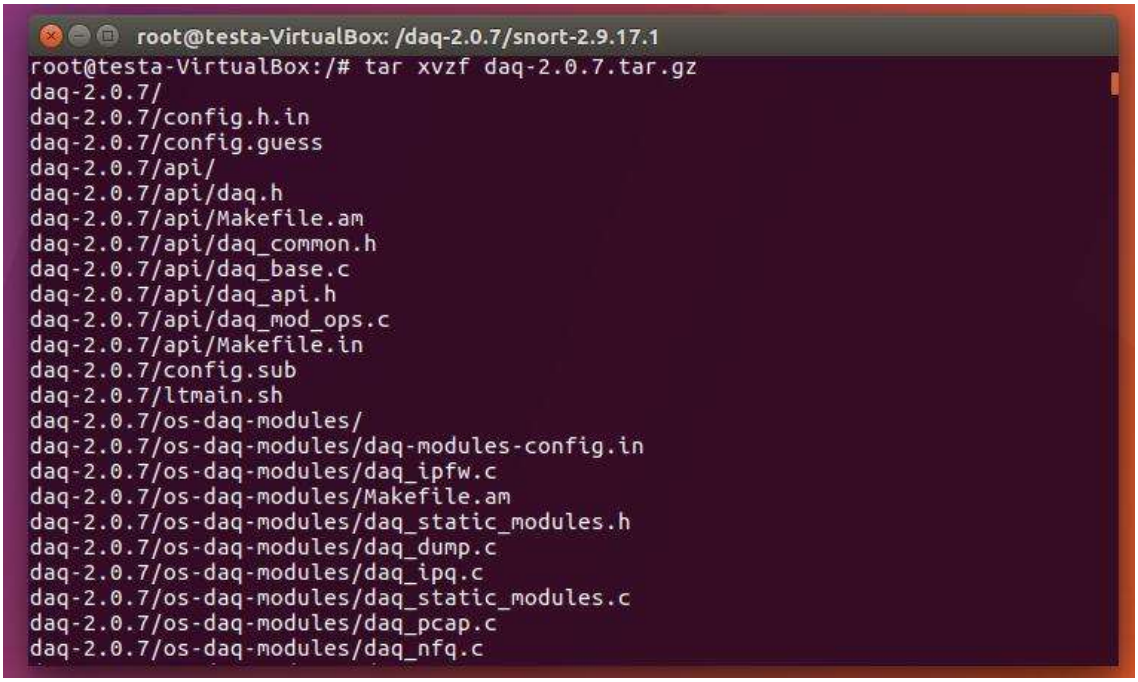


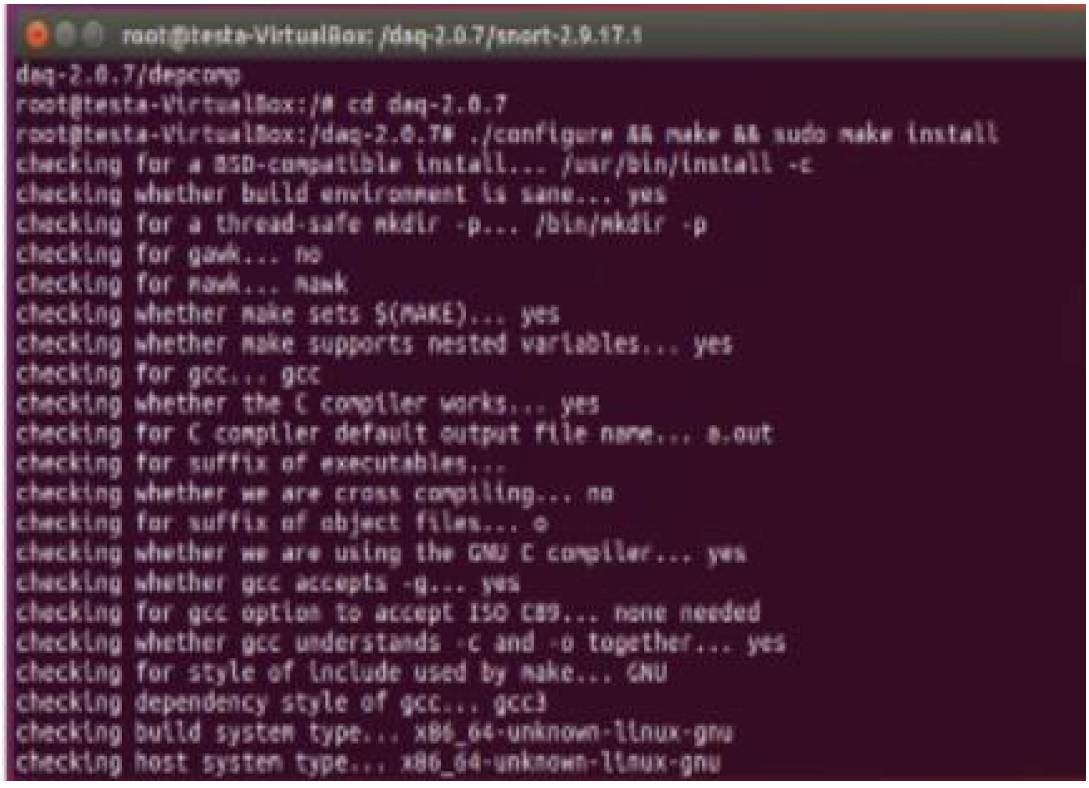
At the beginning, when starting to download the snort, we get a NO CHECK CENRTİFICATE error as follows. This means that the SSL certificate is not valid. In this case, we will try to download the package without ssl verification, so we use the –no-check-certificate

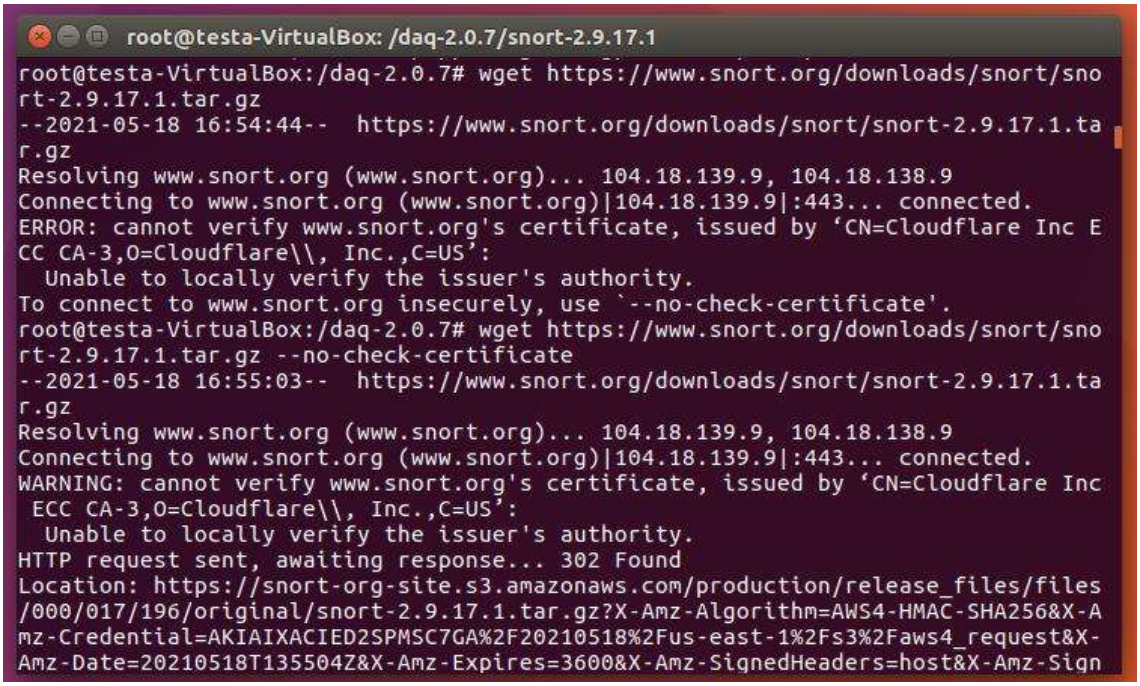
command.

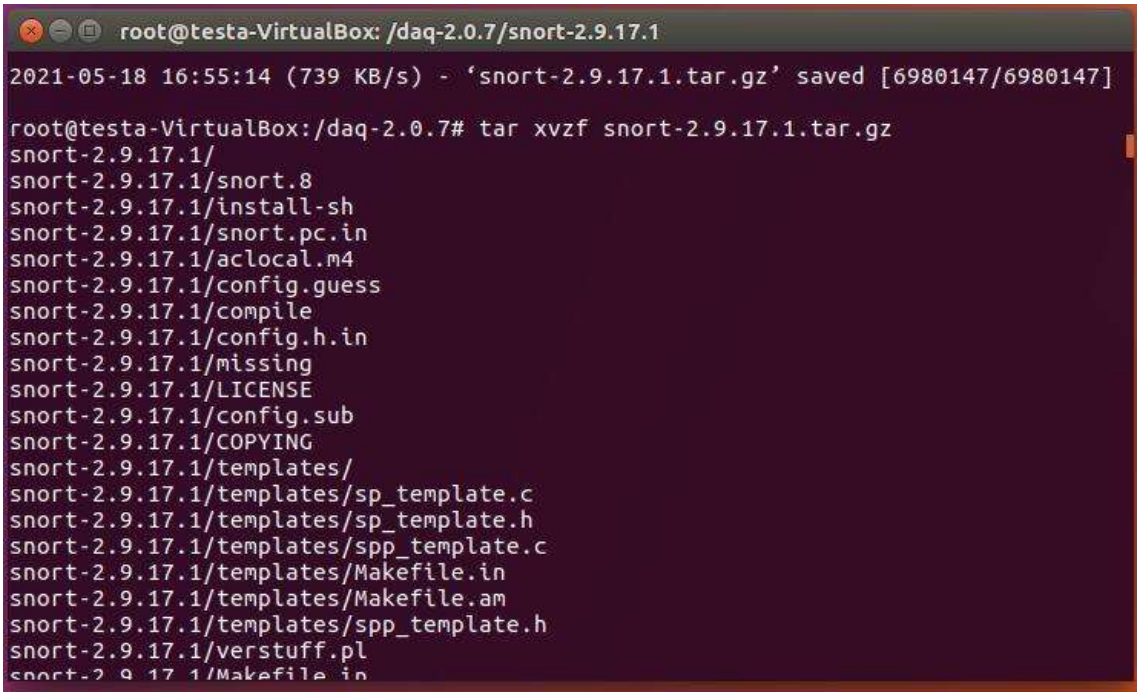

root@testa-VirtualBox: /home/testa
root@testa-VirtualBox:/home/testa# wget https://www.snort.org/downloads/snort/daq-2.0.7.tar.gz
--2021-05-18 15:35:33-- https://www.snort.org/downloads/snort/daq-2.0.7.tar.gz
Resolving www.snort.org (www.snort.org)... 104.18.139.9, 104.18.138.9
Connecting to www.snort.org (www.snort.org)|104.18.139.9|:443... connected.
ERROR: cannot verify www.snort.org's certificate, issued by ‘CN=Cloudflare Inc ECC CA-3,O=Cloudflare\\, Inc.,C=US’:
  Unable to locally verify the issuer's authority.
To connect to www.snort.org insecurely, use `--no-check-certificate'.
root@testa-VirtualBox:/home/testa#


root@testa-VirtualBox: /daq-2.0.7/snort-2.9.17.1
root@testa-VirtualBox:/# wget https://www.snort.org/downloads/snort/daq-2.0.7.tar.gz --no-check-certificate
--2021-05-18 16:52:41-- https://www.snort.org/downloads/snort/daq-2.0.7.tar.gz
Resolving www.snort.org (www.snort.org)... 104.18.138.9, 104.18.139.9
Connecting to www.snort.org (www.snort.org)|104.18.138.9|:443... connected.
WARNING: cannot verify www.snort.org's certificate, issued by ‘CN=Cloudflare Inc ECC CA-3,O=Cloudflare\\, Inc.,C=US’:
  Unable to locally verify the issuer's authority.
HTTP request sent, awaiting response... 302 Found
Location: https://snort-org-site.s3.amazonaws.com/production/release_files/files/000/017/204/original/daq-2.0.7.tar.gz?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIXACIED2SPMSC7GA%2F20210518%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20210518T135242Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=be6938d61552c0ae4f989773799cf23dd7023491bd7c1e0b0078f294e787400b [following]
--2021-05-18 16:52:42-- https://snort-org-site.s3.amazonaws.com/production/release_files/files/000/017/204/original/daq-2.0.7.tar.gz?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIXACIED2SPMSC7GA%2F20210518%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20210518T135242Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=be6938d61552c0ae4f989773799cf23dd7023491bd7c1e0b0078f294e787400b
Resolving snort-org-site.s3.amazonaws.com (snort-org-site.s3.amazonaws.com)... 52.217.65.156
Connecting to snort-org-site.s3.amazonaws.com (snort-org-site.s3.amazonaws.com)|52.217.65.156|:443... connected.


root@testa-VirtualBox: /daq-2.0.7/snort-2.9.17.1
root@testa-VirtualBox:/# tar xvzf daq-2.0.7.tar.gz
daq-2.0.7/
daq-2.0.7/config.h.in
daq-2.0.7/config.guess
daq-2.0.7/api/
daq-2.0.7/api/daq.h
daq-2.0.7/api/Makefile.am
daq-2.0.7/api/daq_common.h
daq-2.0.7/api/daq_base.c
daq-2.0.7/api/daq_api.h
daq-2.0.7/api/daq_mod_ops.c
daq-2.0.7/api/Makefile.in
daq-2.0.7/config.sub
daq-2.0.7/ltmain.sh
daq-2.0.7/os-daq-modules/
daq-2.0.7/os-daq-modules/daq-modules-config.in
daq-2.0.7/os-daq-modules/daq_ipfw.c
daq-2.0.7/os-daq-modules/Makefile.am
daq-2.0.7/os-daq-modules/daq_static_modules.h
daq-2.0.7/os-daq-modules/daq_dump.c
daq-2.0.7/os-daq-modules/daq_ipq.c
daq-2.0.7/os-daq-modules/daq_static_modules.c
daq-2.0.7/os-daq-modules/daq_pcap.c
daq-2.0.7/os-daq-modules/daq_nfq.c


root@testa-VirtualBox: /daq-2.0.7/snort-2.9.17.1
daq-2.0.7/depcomp
root@testa-VirtualBox:/# cd daq-2.0.7
root@testa-VirtualBox:/daq-2.0.7# ./configure && make && sudo make install
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether gcc understands -c and -o together... yes
checking for style of include used by make... GNU
checking dependency style of gcc... gcc3
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu


root@testa-VirtualBox: /daq-2.0.7/snort-2.9.17.1
root@testa-VirtualBox:/daq-2.0.7# wget https://www.snort.org/downloads/snort/snort-2.9.17.1.tar.gz
--2021-05-18 16:54:44-- https://www.snort.org/downloads/snort/snort-2.9.17.1.tar.gz
Resolving www.snort.org (www.snort.org)... 104.18.139.9, 104.18.138.9
Connecting to www.snort.org (www.snort.org)|104.18.139.9|:443... connected.
ERROR: cannot verify www.snort.org's certificate, issued by 'CN=Cloudflare Inc ECC CA-3,O=Cloudflare\\, Inc.,C=US':
Unable to locally verify the issuer's authority.
To connect to www.snort.org insecurely, use `--no-check-certificate'.
root@testa-VirtualBox:/daq-2.0.7# wget https://www.snort.org/downloads/snort/snort-2.9.17.1.tar.gz --no-check-certificate
--2021-05-18 16:55:03-- https://www.snort.org/downloads/snort/snort-2.9.17.1.tar.gz
Resolving www.snort.org (www.snort.org)... 104.18.139.9, 104.18.138.9
Connecting to www.snort.org (www.snort.org)|104.18.139.9|:443... connected.
WARNING: cannot verify www.snort.org's certificate, issued by 'CN=Cloudflare Inc ECC CA-3,O=Cloudflare\\, Inc.,C=US':
Unable to locally verify the issuer's authority.
HTTP request sent, awaiting response... 302 Found
Location: https://snort-org-site.s3.amazonaws.com/production/release_files/files/000/017/196/original/snort-2.9.17.1.tar.gz?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIXACIED2SPMSC7GA%2F20210518%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20210518T135504Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Sign


root@testa-VirtualBox: /daq-2.0.7/snort-2.9.17.1
2021-05-18 16:55:14 (739 KB/s) - 'snort-2.9.17.1.tar.gz' saved [6980147/6980147]
root@testa-VirtualBox:/daq-2.0.7# tar xvzf snort-2.9.17.1.tar.gz
snort-2.9.17.1/
snort-2.9.17.1/snort.8
snort-2.9.17.1/install-sh
snort-2.9.17.1/snort.pc.in
snort-2.9.17.1/aclocal.m4
snort-2.9.17.1/config.guess
snort-2.9.17.1/compile
snort-2.9.17.1/config.h.in
snort-2.9.17.1/missing
snort-2.9.17.1/LICENSE
snort-2.9.17.1/config.sub
snort-2.9.17.1/COPYING
snort-2.9.17.1/templates/
snort-2.9.17.1/templates/sp_template.c
snort-2.9.17.1/templates/sp_template.h
snort-2.9.17.1/templates/spp_template.c
snort-2.9.17.1/templates/Makefile.in
snort-2.9.17.1/templates/Makefile.am
snort-2.9.17.1/templates/spp_template.h
snort-2.9.17.1/verstuff.pl
snort-2.9.17.1/Makefile.in

```
root@testa-VirtualBox: /daq-2.0.7/snort-2.9.17.1
snort-2.9.17.1/doc/README.file_ips
snort-2.9.17.1/configure.in
snort-2.9.17.1/depcomp
snort-2.9.17.1/configure
snort-2.9.17.1/VERSION
snort-2.9.17.1/RELEASE.NOTES
root@testa-VirtualBox:/daq-2.0.7# cd snort-2.9.17.1
root@testa-VirtualBox:/daq-2.0.7/snort-2.9.17.1# ./configure --enable-sourcefire
 && make && sudo make install
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking for style of include used by make... GNU
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
```

12. Configuration of the Snort

To see if the snort is working properly, we use the snort -V command to check the version. If there is a screen like the one below, the snort setup is completed successfully.

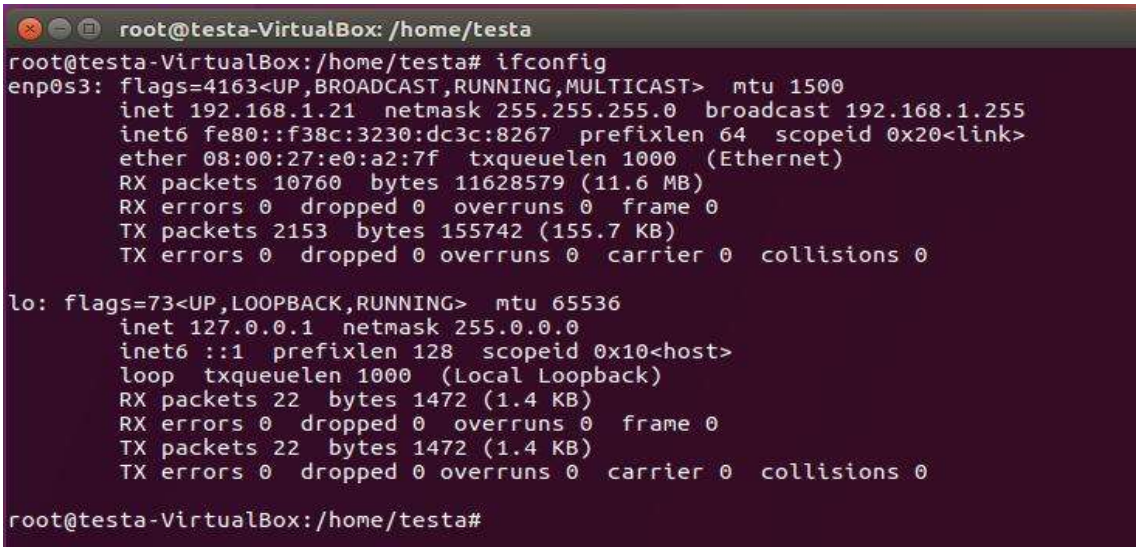
```
root@testa-VirtualBox: /
root@testa-VirtualBox:/# snort -V

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.7.0 GRE (Build 149)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.8.1
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.11

root@testa-VirtualBox:/#
```

After completing the installation, we first begin to complete the configuration settings. Since we want to run it at normal user level, we do not make much separation. we only change the ip address. We run the ifconfig command.

We need to configure our HOME_NET value: the network we will be protecting. First, we enter ifconfig in your terminal shell to see the network configuration. We note the IP address and the network interface value.

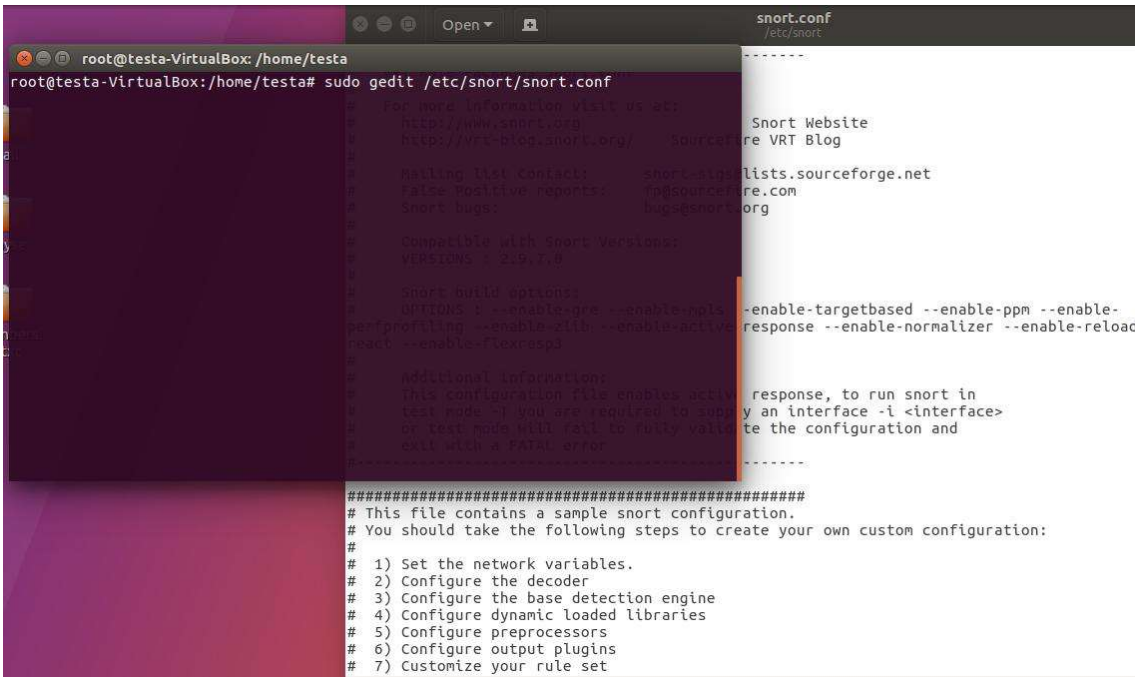
Our ip address is 192.168.1.21 and our network card name is enp0s3.

```
ifconfig
```

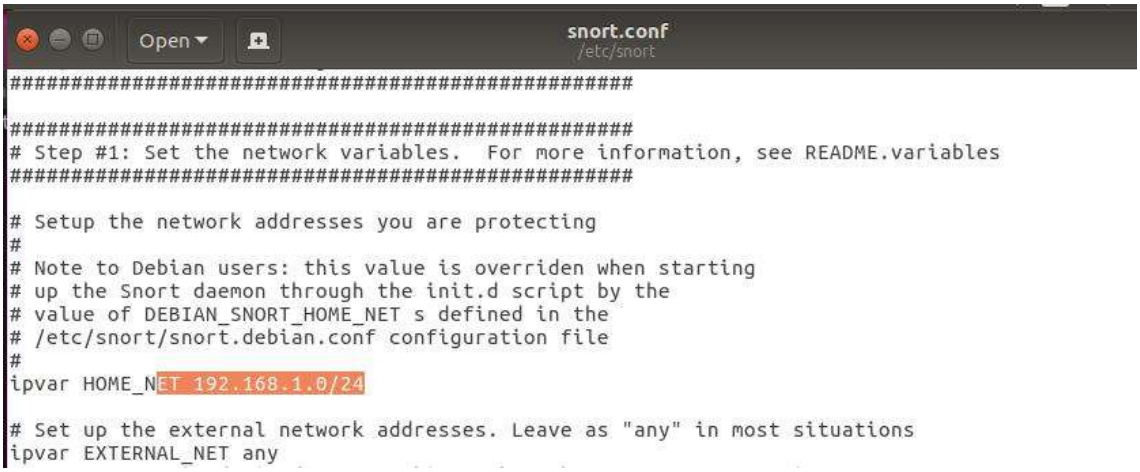


After, we type the following command to open the snort configuration file in gedit text editor:

```
sudo gedit /etc/snort/snort.conf
```



When the snort.conf file opens, scroll down until we find the ipvar HOME_NET setting. We was want to change the IP address to be our actual class C subnet. Currently, it should be 192.168.1.0/24. We simply changed the IP address part to match your Ubuntu Server VM IP, making sure to leave the ".0/24″ on the end.



We selected Save from the bar on top and close the file. At this point, Snort is ready to run. Except, it doesn't have any rules loaded. To verify, we run the following command:

```
sudo snort -T -i eth0 -c /etc/snort/snort.conf
```

Here we are telling Snort to test (-T) the configuration file (-c points to its location) on the enp0s3 interface This will produce a lot of output.

```
root@testa-VirtualBox: /

root@testa-VirtualBox:/# sudo snort -T -i eth0 -c /etc/snort/snort.conf
Running in Test mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
PortVar 'HTTP_PORTS' defined :  [ 80:81 311 383 591 593 901 1220 1414 1741 1830
2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7000:7001 7144:7145 7510 7777
7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 8243 8280 8300
8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:34444 41080 50002
55555 ]
PortVar 'SHELLCODE_PORTS' defined :  [ 0:79 81:65535 ]
PortVar 'ORACLE_PORTS' defined :  [ 1024:65535 ]
PortVar 'SSH_PORTS' defined :  [ 22 ]
PortVar 'FTP_PORTS' defined :  [ 21 2100 3535 ]
PortVar 'SIP_PORTS' defined :  [ 5060:5061 5600 ]
PortVar 'FILE_DATA_PORTS' defined :  [ 80:81 110 143 311 383 591 593 901 1220 14
14 1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7000:7001 7144:71
45 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 82
43 8280 8300 8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:34444
 41080 50002 55555 ]
```

By typing snort -r, we can see the list of commands we can use in snort.

```
snort -r
```

```
root@testa-VirtualBox:/# snort -r
snort: option requires an argument -- 'r'

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.7.0 GRE (Build 149)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.8.1
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.11

USAGE: snort [-options] <filter options>
Options:
        -A          Set alert mode: fast, full, console, test or none  (alert file alerts only)
                    "unsock" enables UNIX socket logging (experimental).
        -b          Log packets in tcpdump format (much faster!)
        -B <mask>   Obfuscated IP addresses in alerts and packet dumps using CIDR mask
        -c <rules>  Use Rules File <rules>
        -C          Print out payloads with character data only (no hex)
        -d          Dump the Application Layer
        -D          Run Snort in background (daemon) mode
        -e          Display the second layer header info
        -f          Turn off fflush() calls after binary log writes
        -F <bpf>    Read BPF filters from file <bpf>
        -g <gname>  Run snort gid as <gname> group (or gid) after initialization
        -G <0xid>   Log Identifier (to uniquely id events for multiple snorts)
        -h <hn>     Set home network = <hn>
                    (for use with -l or -B, does NOT change $HOME_NET in IDS mode)
        -H          Make hash tables deterministic.
        -i <if>     Listen on interface <if>
        -I          Add Interface name to alert output
        -k <mode>   Checksum mode (all,noip,notcp,noudp,noicmp,none)
        -K <mode>   Logging mode (pcap[default],ascii,none)
        -l <ld>     Log to directory <ld>
        -L <file>   Log to this tcpdump file
        -M          Log messages to syslog (not alerts)
        -m <umask>  Set umask = <umask>
        -n <cnt>    Exit after receiving <cnt> packets
        -N          Turn off logging (alerts still work)
        -O          Obfuscate the logged IP addresses
```

Now the snort is ready to use and we can start the snort and start detecting attacks. We will do several examples to show how snort works. First we will try to send the ping and how the snort notices this by running the snort in different alerts modes. Then, We will create a new

rule to detect ftp connections and test it. We will also try to send a ping by writing a rule. Next we will look at the nmap scan and namp tcp scan jobs.

13. Detecting Ping in Snort With Various Snort Alerts Modes

Now we will Snort using different alerts modes, send pings and see what happens. The difference of these alerts modes from each other is that their console output, ie their appearance, is different.

13.1. CMG Alert Example

Now let's get a report in the console with the information of a full report and more by using the command below.

```
snort -c /etc/snort/snort.conf -q -A cmg
```

snort= calls the program

-c= path to config file, in this case the default one (/etc/snort/snort.conf)

-q= prevents snort from displaying initial information

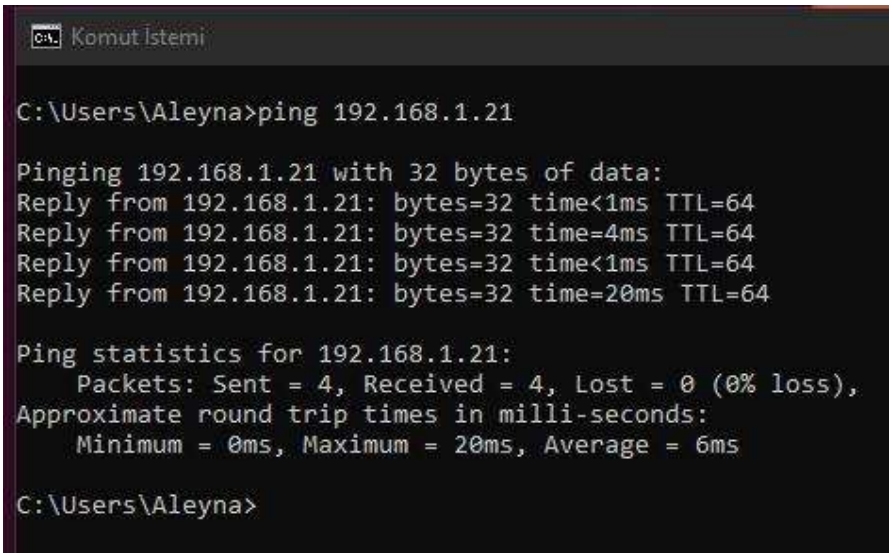
-A= defines the alert mode, in this case cmg.

While running this mode, although we did nothing before sending ping, Snort detects bad traffic on the network as follows. So we understood how snort actually works.

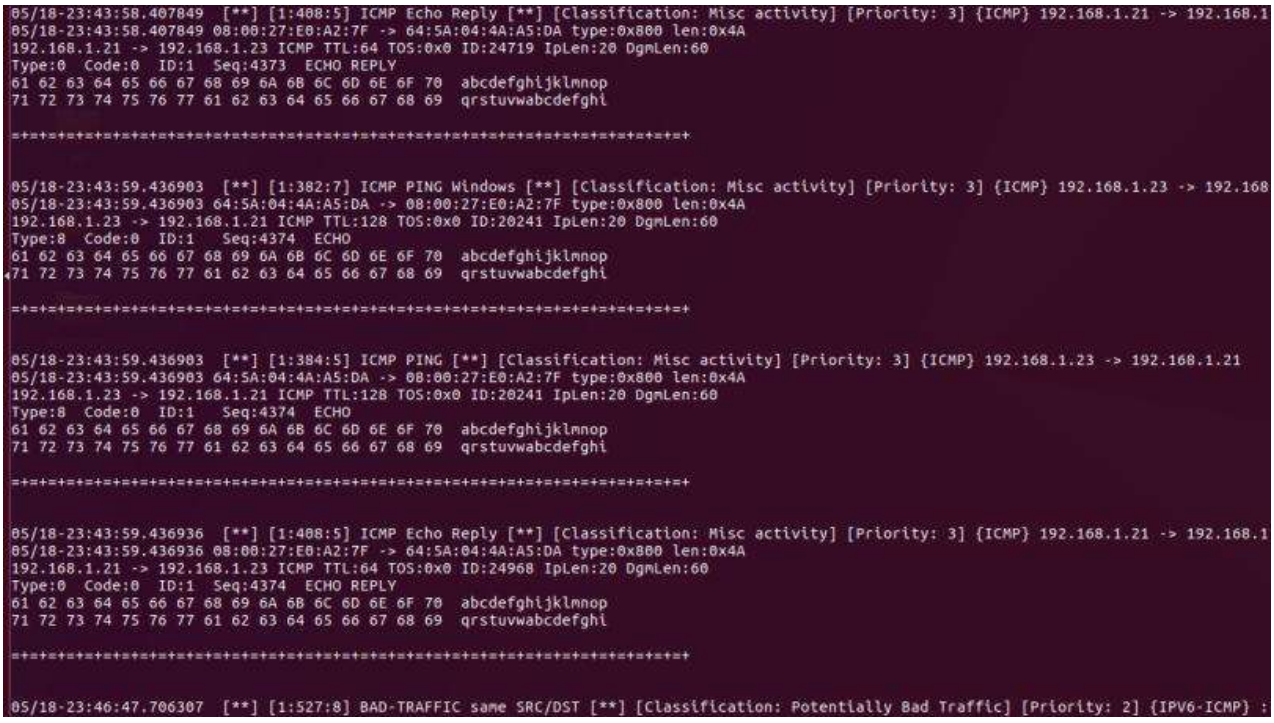
```
root@testa-VirtualBox:/# snort -c /etc/snort/snort.conf -q -A cmg
05/18-23:42:53.213566  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially
 Bad Traffic] [Priority: 2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
05/18-23:42:53.213566 64:5A:04:4A:A5:DA -> FF:FF:FF:FF:FF:FF type:0x800 len:0x16C
0.0.0.0:68 -> 255.255.255.255:67 UDP TTL:128 TOS:0x0 ID:7444 IpLen:20 DgmLen:350
Len: 322
01 01 06 00 F0 59 4F A6 00 00 00 00 00 00 00 00  .....YO.........
00 00 00 00 00 00 00 00 00 00 00 00 64 5A 04 4A  ............dZ.J
A5 DA 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 63 82 53 63  ............c.Sc
35 01 03 3D 07 01 64 5A 04 4A A5 DA 32 04 C0 A8  5..=..dZ.J..2...
01 17 0C 0F 44 45 53 4B 54 4F 50 2D 55 44 42 50  ....DESKTOP-UDBP
43 32 42 51 12 00 00 00 44 45 53 4B 54 4F 50 2D  C2BQ....DESKTOP-
```

First, we ping our virtual machine over Windows 10 with the following command.

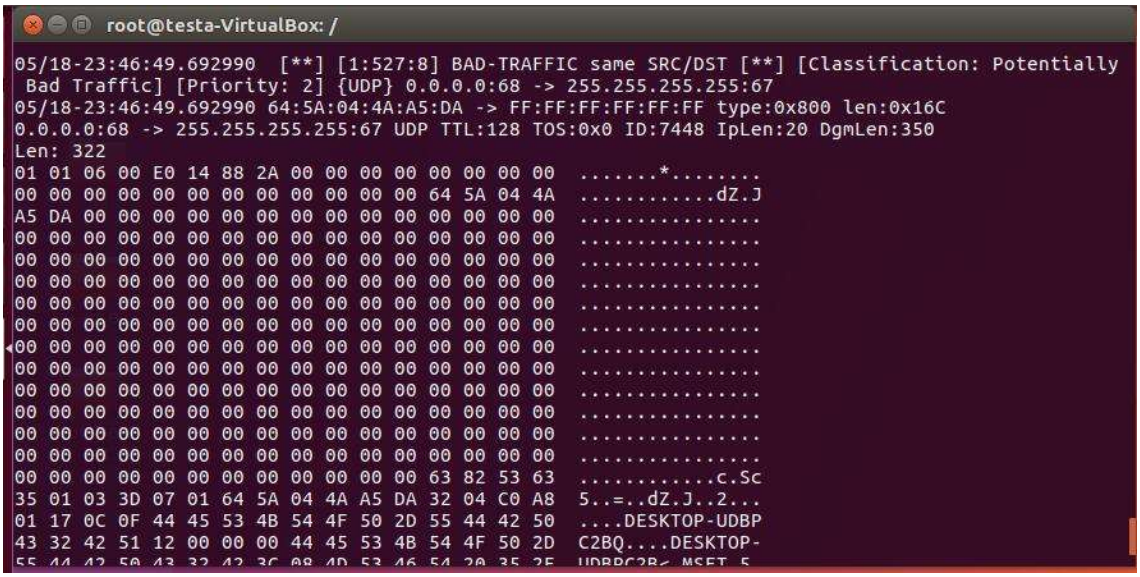
```
Ping 192.168.1.21
```



After sending the ping, it detects the ping as follows. The ICMP ping is displayed on the screen as windows. We understand that there is a ping from the windows operating system. It's as if classfication has classified it as miss activity. This is important, since we run the snort in cmg mode, this alert appears on the screen is different, so the console view is different.



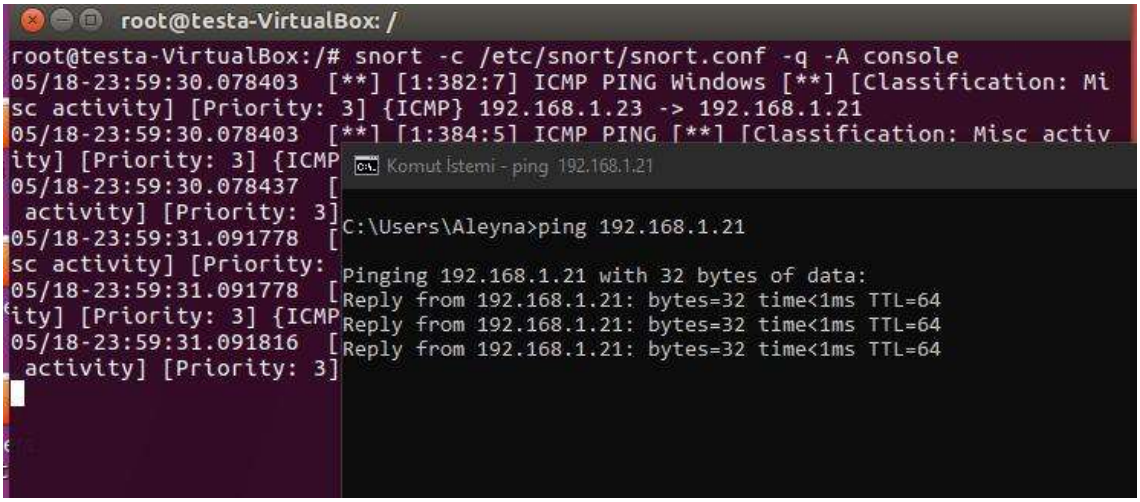
Even though the ping is over, the snort has detected another bad traffic.



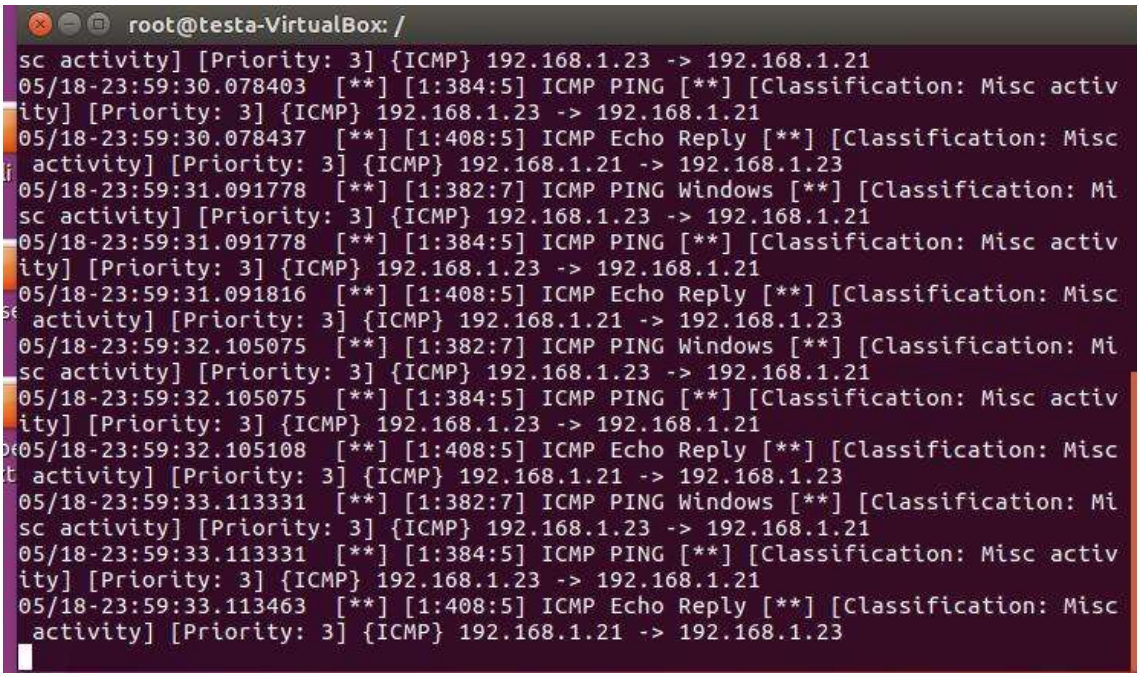
13.2.Console Alert Test

Console: prints fast alerts in the console. (-A console)

We are now pinging in console alerts mode.



Here ICMP Ping shows that windows have come from 192.168.1.23 to 21 its priorty is 3 and classification is mis activity.



Now we will write our own rule and detect the ping by snort that way.

First, we open the file with the following command to write a rule.

```
Sudo gedit /etc/snort/rules/local.rules
```

Then we created our own ping rule

alert – Rule action. Snort will generate an alert when the set condition is met.

any – Source IP. Snort will look at all sources.

any – Source port. Snort will look at all ports.

-> – Direction. From source to destination.

$HOME_NET – Destination IP. We are using the HOME_NET value from the snort.conf file. 192.168.1.0/24

any – Destination port. Snort will look at all ports on the protected network.

Rule options:

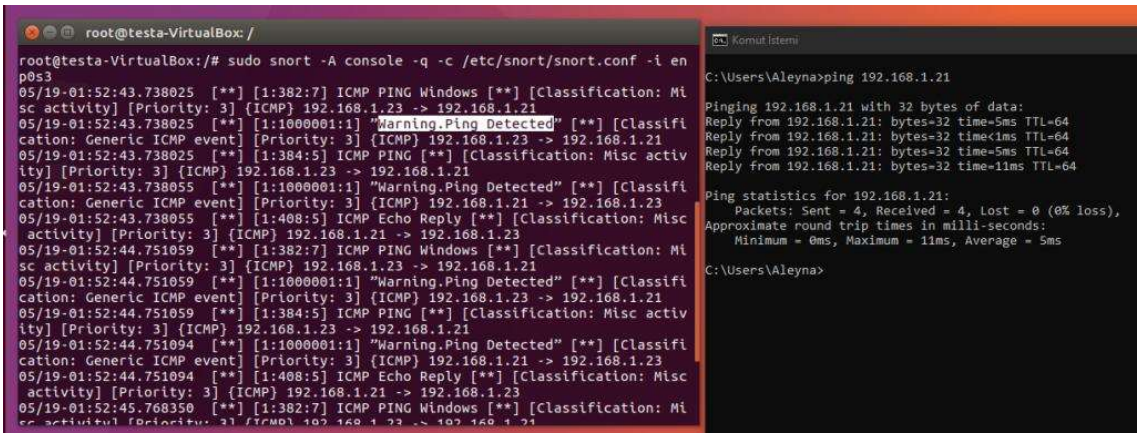
msg:” Warning ping detected.”

sid:1000001 – Snort rule ID.

rev:1 – Revision number. This option allows for easier rule maintenance.

classtype:icmp-event – Categorizes the rule as an “icmp-event”, This option helps with rule organization.

Then we send the ping and run the snort and see what happened. Our rule ran snort and wrote Warning ping Detected on the
screen.



14. Snort FTP Example

As in the example below, if there is an ftp connection to our ip address, snort can detect it.

In Snort, everything proceeds based on rules. We will write kyaral here and if there is such an attack, the snort will give alerts as below and the screen will be written FTP CONNECTION ATTEMT.

alert ---> show alert

TCP ---> It's a protocol Transmission Control Protocol

-> :- to

192.168.1.0/24 ---> destination ip

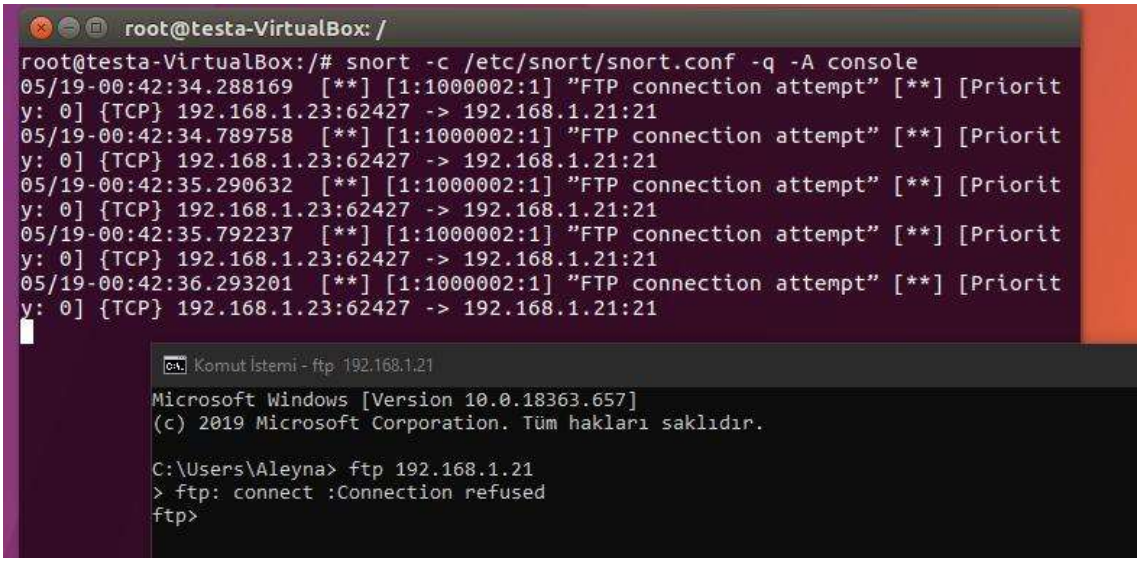
msg ---> shows message which we write

sid ---> keyword is used to uniquely identify Snort rules. This information allows output plugins to identify rules easily.

100 - 1,000,000 Rules already registered . So u need to use greater than this id like 1,000,123.

rev ---> keyword is used to uniquely identify revisions of Snort rules

We establish a Windows ftp connection, for this we write ftp 192.168.1.21 in the command system in windows.

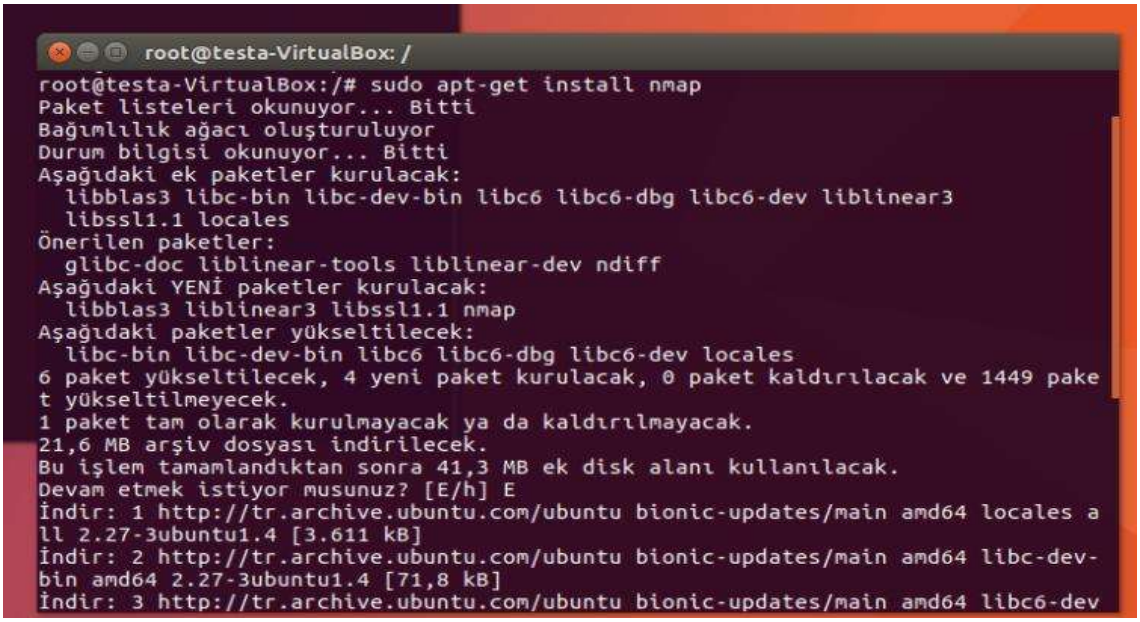


15. Snort and Nmap Examples

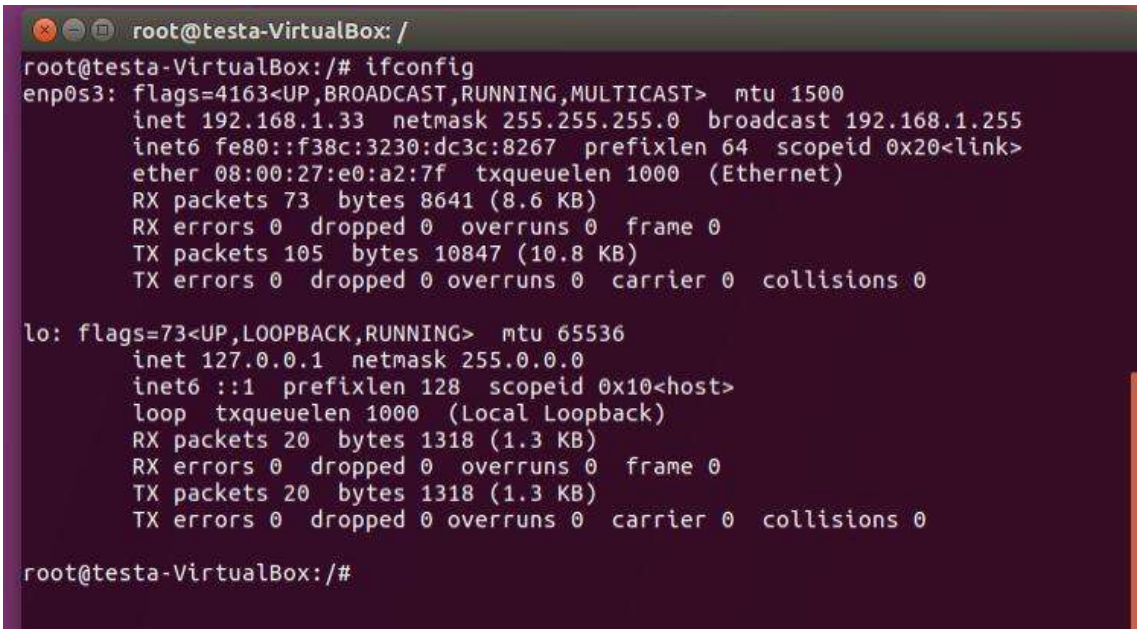
We will install nmap on the same operating system, scan the network and watch it in the snort.

First, we download the nmap with the command below.

```
sudo apt-get install nmap
```

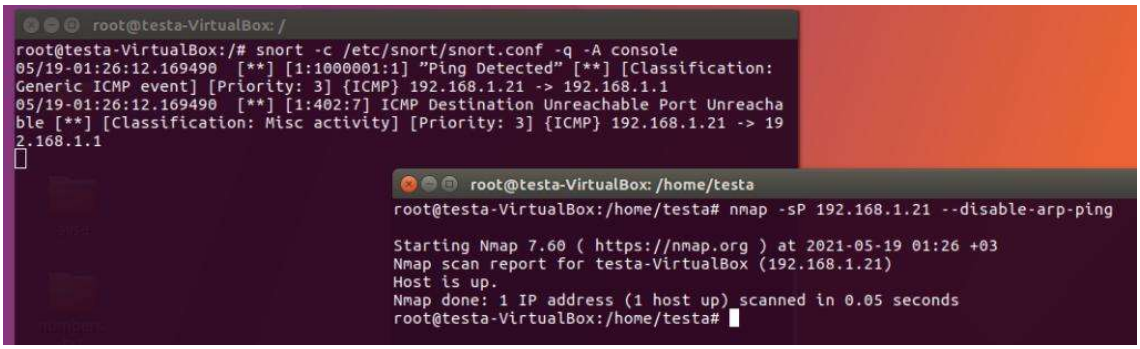


We check our ip address again with the ifconfig command and continue with our 192.168.1.33 ip address below.



Without writing any rules, we ran the Snort in console mode and scanned the nmap using the command below.

```
nmap -sP 192.168.1.21 --disable-arp-ping
```



Before writing our own nmap rules, we opened our rule file with the following command.

```
Sudo gedit /etc/snort/rules/local.rules
```

Now we will do the same by writing our own rule.

alert ---> show alert

ICMP ---> It's a protocol used to report error in ipv4

-> :- to

$HOME_NET 22 ---> destination ip

msg --->Nmap Scan Detected

sid ---> keyword is used to uniquely identify Snort rules. This information allows output plugins to identify rules easily.

rev ---> keyword is used to uniquely identify revisions of Snort rules

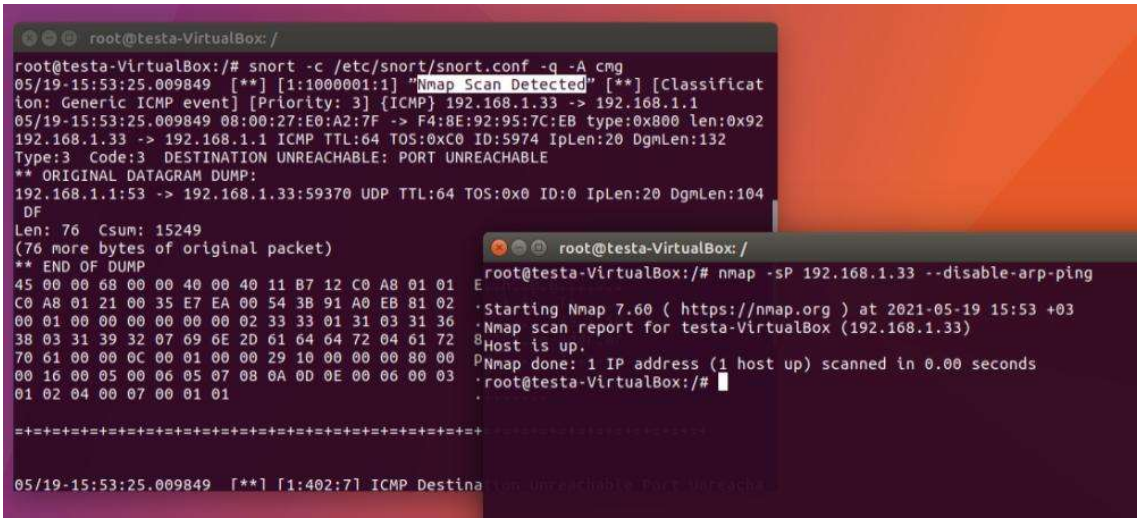
classtype:icmp-event ---> Categorizes the rule as an "icmp-event", one of the predefined Snort categories.

We are running our snort now with the command below comandin alerts mode at the top in cmg mode. console looks like in cmg mode as below.

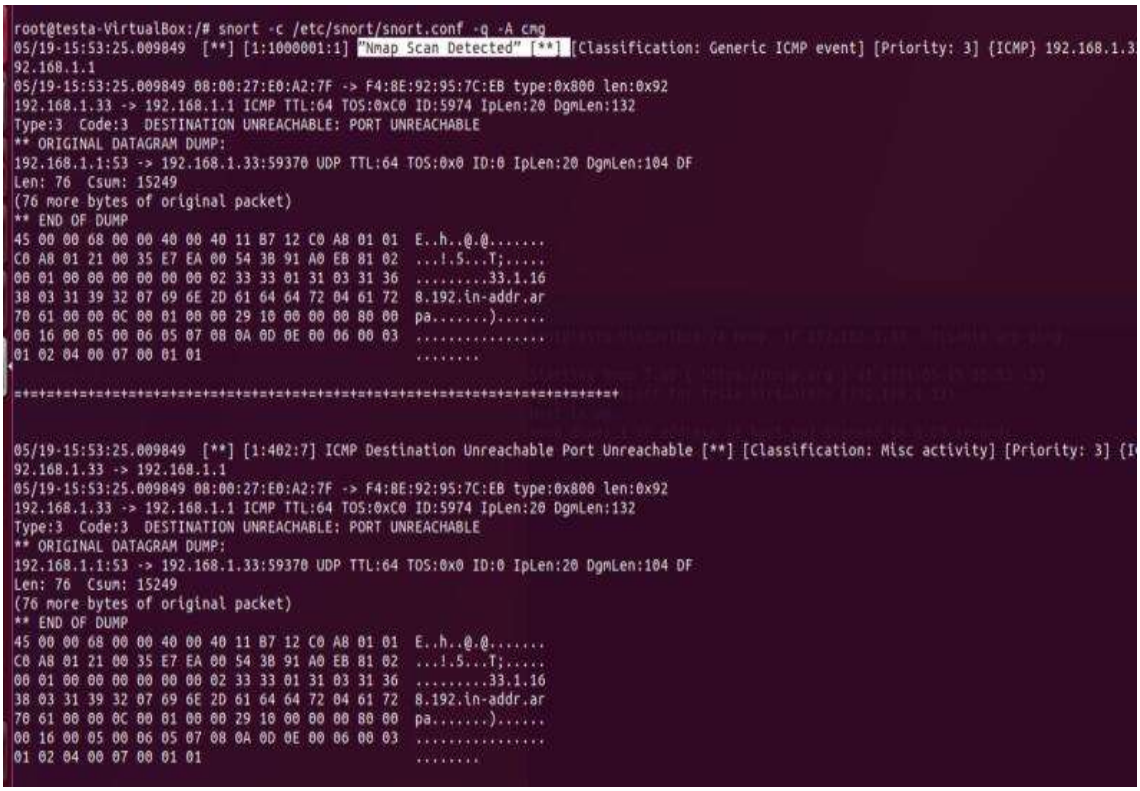
```
Snort -c /etc/snort/snort.conf -q -A -cmg
```

We did an nmap scan with command below and the snort console also used our rule and wrote nmap scan detected on the screen.

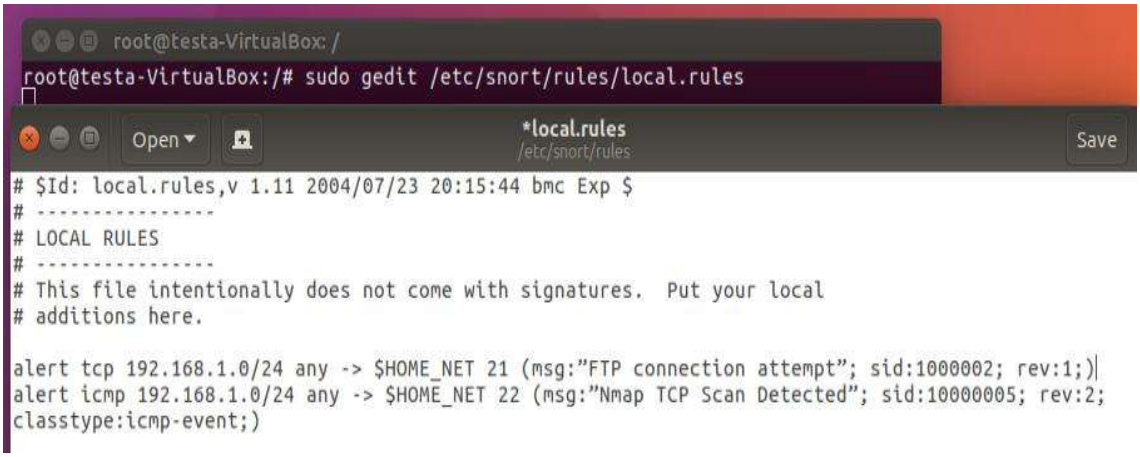
```
Nmap -sp 192.168.1.33 –disable.arp.ping
```



More detailed picture



We're doing our second nmap example with TCP.We write our own rule and scan for TCP with nmap. We opened a file and wrote our rule.



alert ---> show alert

TCP ---> It's a protocol used to report error in ipv4

-> :- to

$HOME_NET 21 ---> destination ip

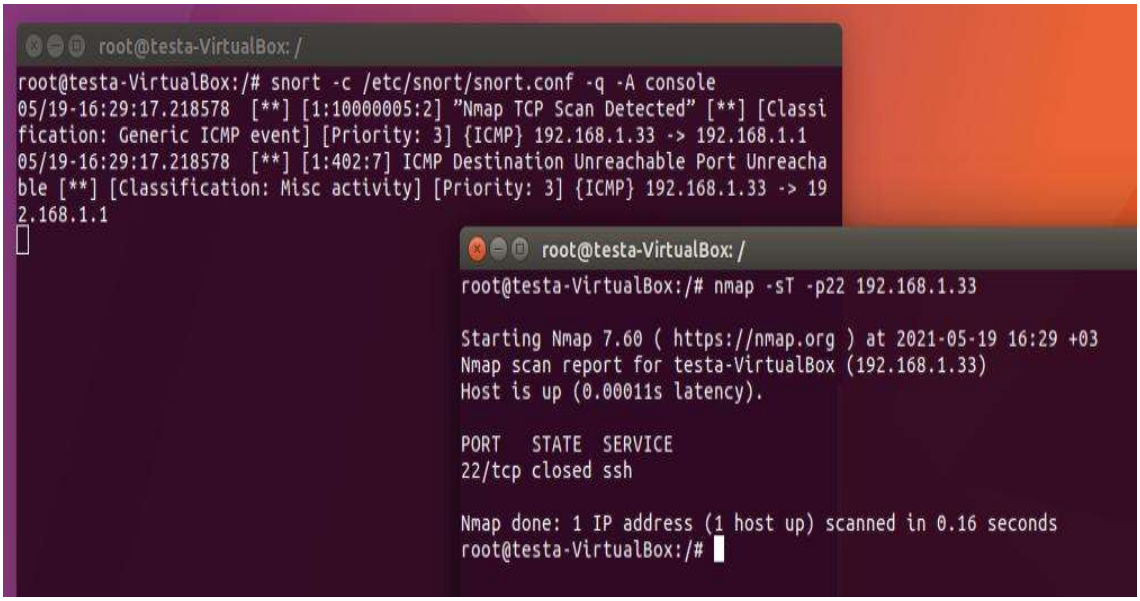
msg ---> FTP connection attempt

sid ---> keyword is used to uniquely identify Snort rules. This information allows output plugins to identify rules easily.

rev ---> keyword is used to uniquely identify revisions of Snort rules

classtype:tcp-event ---> Categorizes the rule as an “tcp-event”, one of the predefined Snort categories.

We ran snort in Console mode and nmap detected it saying tcp scan detected.



16.Conclusion

Snort is a cyber security solution that detects network intrusion attempts and vulnerabilities on the network. Snort looks at network traffic in real time, streamlines security and authentication efforts, and performs detailed analysis. In this case, it is an important tool used by Cyber Security Specialists. Snort security mode packet sniffer has 3 different modes, such as consistent logging of network traffic and active network attack to facilitate debugging. Additionally, Snort was created to detect various types of attacks and uses a flexible rule language to determine the types of network traffic that needs to be collected. As far as we've learned while implementing, snort is completely dependent on rule structure. These rules are used to find packages that match them and make necessary warnings to system users. Thusly, for Snort to work properly, the user must use commands that will define the rules to use and specify how the program should operate in one of the three basic modes. Actuaaly, we can say that snort is made very simply because it offers aesthetic convenience to the user and supports it with different alert modes. For example, we did not encounter any problems while using Snort after installing it properly because it has a very simple interface and we were able to learn and use it easily. However, the process of setting up and configuring the Snort has a real challenge. We encountered an access block and no certificate error while we were setting up and correcting the initial settings. Therefore, our recommendation to snorta is to support the user to facilitate this installation phase and ilj configuration. For example, your website conveys quite a bit of incomplete information on these issues. In addition, Snort provides us a lot of convenience. Örenğin Snort analyzes all connections made on IP networks and keeps them in the Log (Entry, Exit) records. In addition, Snort Content, which contains many software modules, offers many support such as CGI attacks, port scanning, buffering to the user. But it also finds a downside, as we investigated. For example, it offers fewer features against paid cyber security solutions. In this case, our recommendation is to use the Snort tool supported by additional programs and graphics.As a result, Snort can be preferred as a simple and effective cyber security solution for individuals like us who have just started learning in the field of cyber security.

17. References

- Raj Chande, December 22, 2017, https://www.hackingarticles.in/detect-nmap-scan-using-snort/
- Infosec, March 1, 2021, https://resources.infosecinstitute.com/topic/snort-rules-workshop-part-one/
- sankethj, 3 Ara 2020, https://dev.to/sankethj/detect-dos-ping-etc-using-snort-4gab
- Ivan Vanney, 2019, https://linuxhint.com/snort_alerts/
- Hüsnü İŞLEYEN, 8 Sep 2014, https://github.com/slyn/Snort/blob/master/Snort%20ve%20Eklentileri.md
- Prısma, https://www.prismacsi.com/snort-kullanimi/#:~:text=Snort%20esnek%20mimarisi%20sayesinde%20bir,esnek%20kurallar%20yazabilme%20imkan%C4%B1%20vermesidir.
- P. Israelsson, J.Karlsson and G. Giamarchi, October 17, 2005, https://www.it.uu.se/edu/course/homepage/sakdat/ht05/assignments/pm/programme/Introduction_to_snort.pdf