Department of Informatics

SNORT LAB

Network Security

Diego Sempreboni

INTRODUCTION

Snort is a versatile tool that not only allows system administrators to define network firewall rules to protect a network, but it also enables them con plug IDS-like (Intrusion Detection System) capabilities on the target network node. In particular, Snort allow is capable of detecting both signatures and anomalies in the header of a network packet. Snort is an open-source tool that can run on top of multiple platforms such as Linux, Windows, Aix, Solaris, etc.

The installation folder of Snort might change across platforms and in different distributions. In Linux systems it is usually kept in a file called **/etc/snort/snort.conf**. Snort process incoming traffic and looks for patterns that match against pre-defined rules. These rules are grouped into different categories and they are usually stored in different files according to the category. Processing packets and mapping them against the existing rules is a costly task. The performance of Snort directly depends on the number of existing rules.

NOTE: You need to be **sudo** to run Snort.

SNORT KEY COMPONENTS

The architecture of Snort contains several components that work together to detect potential intrusions. These components are:

- **Packet decoder**: collects the packets from the different network interfaces and feeds them to the component called *pre-processor*, which it is explained next.
- **Pre-processor**: this is an engine that accepts a number of plug-ins that can be used to process and also modify the packets before they are sent to the next component.
- **Detection engine**: this is the core element of Snort, which uses pre-defined rules to detect signatures within the network packets.
- **Alert System**: alerts are by default store in a directory, typically located in **/var/log/snort**. These are stored both in clear text and binary format.
- **Output module**: These are plug-ins that perform different functions depending on how you want to store the logs and alarms. They basically control the type of output.

Task: Open the aforementioned configuration file and explore existing rules.

BASTIONING THE SNORT

This first part of the lab aims at re-visiting some of the concepts we've seen in previous sessions. In particular, these are some of the actions we can do when protecting the IDS:

- Avoid having other network services running on the IDS.
- Avoid responding to network scanning protocols (e.g., ping).

Task: Configure the firewall to bastion the IDS in the most restrictive setting.
Question: Can Snort still analyse traffic data?

Other good practices revolve around:

- Having a dedicated system for hosting the IDS.
- Having a system with automated updates.
- Having a system without superfluous services.
- Create only the strictly necessary user accounts.

SNORT BASICS

The next part of the lab aims at walking through the basics of Snort. Read through and follow these steps:

1.- Verify that Snort is working by running in sniffer mode:

```
# snort -dvi eth0
```

This should show all the packets that are running through the network.

2.- Start and stop Snort

It can also be started with the command (use option -c to point to a configuration file):

```
# snort -A full -c /etc/snort/snort.conf
```

The -c option displays verbose information related to the specifics of the configuration loaded from config file (i.e., snort.conf).

To verify that the process has started executing the command:

```
# ps -ef | grep snort
```

3.- Get basic help

You will get help executing the following command:

```
# snort -?
```

You will get the basic help that informs you of the main options with which you can start the program.

Task: Take your time exploring the different options.

4.- Check the operation of snort

Once Snort is up and running, you will be able to check if the captured data is registering any activity. The alerts are stored in the file **/var/log/snort/alert**.

To check if alerts are being stored, view the contents of the file with the command:

```
# more /var/log/snort/alert
```

Alternatively, you can keep dump only the latest alerts with:

```
# tail -f /var/log/snort/alert
```

5.- Use the Sniffer Mode

Snort can operate as regular network sniffer, capturing network packets running through the network. You can use Snort in sniffer mode by using the command:

```
# snort -v
```

This command will display all captured packets on the terminal and can be terminated with *ctrl.-C*. At the end, it shows statistics of the analysed packets.

To display all available information use the *-de* option, and to store the information in a directory use the option *-l <directory>*.

```
# snort -dev -l <directory>
```

This option shows all the information captured using both hexadecimal and ASCII formats.

6.- Definition of Rules

All rules in Snort have two logical parts:

Heather	Options

The **header** contains the *criteria* that should match against the packet to match with the specifications of the rule. It should also define the *action* that should be executed when this happens.

The **options** contain a verbose description of the alert, which will then be logged when the rule is triggered. This should be informative, as it will be used to alert the system administrator about a given attack. It also enables us to provide additional specifications to the rule.

The general structure of the header is as follows:

Action	Protocol	Address	Port	Direction	Address	Port

- **Action**: type of action taken when the criteria are met. Can be:
    - Pass
    - Log
    - Alert
    - Activate
    - Dynamic
- **Protocol**: IP, ICMP, UDP ...
- **Address**: source and destination IP addresses.
- **Port**: source and destination ports of communication.
- **Direction**: Determines which port and address are the origin and which destination.

Example

```
alert ip any any -> any any (msg: "IP packet detected";)
```

The rules are stored in files named <name> .rules that are located in the **/etc/snort** directory. These rules are loaded during the initialization. Thus, the Snort service should be restarted

after every modification. Note that to include rules, the **#include** statement must be declared in the **snort.conf** file pointing to the file where the rules are defined. The *local.rules* file can be used to test rules. Each rule must have a **unique identifier** (use `sid:<number>;` in the options).

ADVANCED SNORT

Options: There are 24 options, the most important ones are:

msg: Message to be reported should the alert is triggered.

dsize: Used to check the size of the payload of a certain packet. Useful, for example, to recognize buffer overflow attacks that send packets larger than a certain buffer size.

content: The most used and most important option. Search for specific contents within the packet's payload. It uses the Boyer-Moore algorithm to deep inspect the packets. It is very slow and should be avoided in favour of other options. It is "case sensitive" and admits both binary (specified between "|" as in the previous example) and alphanumeric data.

```
alert tcp any any -> any 80 (content: "GET"; msg: "GET access";)
```

offset: It is a modifier of the content's rules used to optimize the performance – it alters the initial position of the search.

depth: It is a modifier of the content's rules used to optimize the performance – it puts a limit at the beginning of the searched pattern, making the search more efficient.

nocase: Disable the "case sensitivity" which is the default option in a rule content.

```
alert tcp any any -> any 80 (content: "GET"; msg: "GET access";
nocase;)
```

flags: They can take the following values

F - FIN (LSB in TCP Flags byte)
S - SYN
R - RST
P - PSH
A - ACK
U - URG

Logical operators:

* - ANY flag, any of the specified flags

! - NOT flag, any unspecified

Examples

```
alert any any -> any any (flags: SF; msg: "Possible SYN FIN scan";)
```

```
alert tcp any any -> any any (msg:"TAGGED"; content:"GET"; tag:host,3,packets,src; sid:10001;nocase)
```

```
alert tcp any any -> any 21 (content: "USER root"; nocase; msg: "FTP root user access attempt";)
```

Task: Create a rule that logs all traffic HTTP connections using Javascript executing *alert* function, widely used in many XSS attacks.

Hint: look at the *content* option.

DETECTING A DOS ATTACK

Revisit the lab on *Denial of Service and IP Spoofing* you did a few weeks back and create a Snort rule that would block the DoS attack.

Hint: look at the detection_filter option.

COMMUNITY RULES

As you probably have realized already, the most important element of an Intrusion Detection System is the intelligence you feed it in with. Snort provide a large dataset of rules that are constantly updated. However, you need to be a registered user or have a paid subscription to download these rules.

There is an open community that makes rules available. Note that community rules are not provided in this lab, but one can find them available online:

https://snort.org/downloads/community/snort3-community-rules.tar.gz

https://snort.org/downloads/community/community-rules.tar.gz

REVISION IPSEC

The second part of this lab is dedicated to review possible exam questions about IPsec and TLS. Revisit the material and reason about to the following coursework questions:

- Which are the 3 main subprotocols of IPsec and what security properties do they provide?
- Describe at least 3 benefits of using IPsec.
- In the context of IPsec, what is a Security Association?
- What is the role of the databases SAD and SPD in IPsec?
- Describe how IPsec processes outbound and inbound traffic.
- Describe how the protocol AH works in transport mode and how it works in tunnel mode.
- Describe how the protocol ESP works in transport mode and how it works in tunnel mode (consider only the case of IPv4).
- What is Perfect Forward Secrecy and how can it be achieved?
- What are the main objectives of the IKE protocol?