CO892 - Snort Intrusion Detection System Report

Bastien Dhiver – bfrd2

Table of Contents

CO892 - Snort Intrusion Detection System Report....1
1. Overview....1
2. Detection Rules....2
3. PCRE Matching....3
4. Example rules....4
5. References....4

1. Overview

In this section we will explore the aims that the Snort software attempts to address, its internal architecture and how we can get started.

“[Snort] is an open source intrusion prevention system capable of real-time traffic analysis and packet logging.” [1]

Snort belongs to the Network Intrusion Prevention System (NIPS) category. Cisco is behind this project since 2013. Snort was designed to be a packet sniffing “lightweight” Network Intrusion Detection System (NIDS). It is cross-platform, have a low memory and CPU footprint and is easily configurable in part due to its modular architecture.

The software can be configured in tree different modes:

- Sniffer mode: This mode will display the packets that are read from the network on the console. The use cases are similar to those of the tcpdump tool (libpcap-based). It can be useful to get a quick idea of what is happening in real-time on the network. A filtering interface (BPF) is available to reduce the amount of displayed information.
- Packet Logger mode: The network packets can be written to the disk in several formats (the most famous one would be PCAP). The capture file can then be shared and the analysis be carried out later.
- NIDS mode: detection and analysis will be performed on the incoming network traffic (the selected network interface). Multiple detection modes are available via rules/plugins and are based on signatures, statistical anomaly and protocol verification. Features like TCP stream reassembly and application layer analysis are embedded in the software.

Here is a representation of the Snort internal architecture:

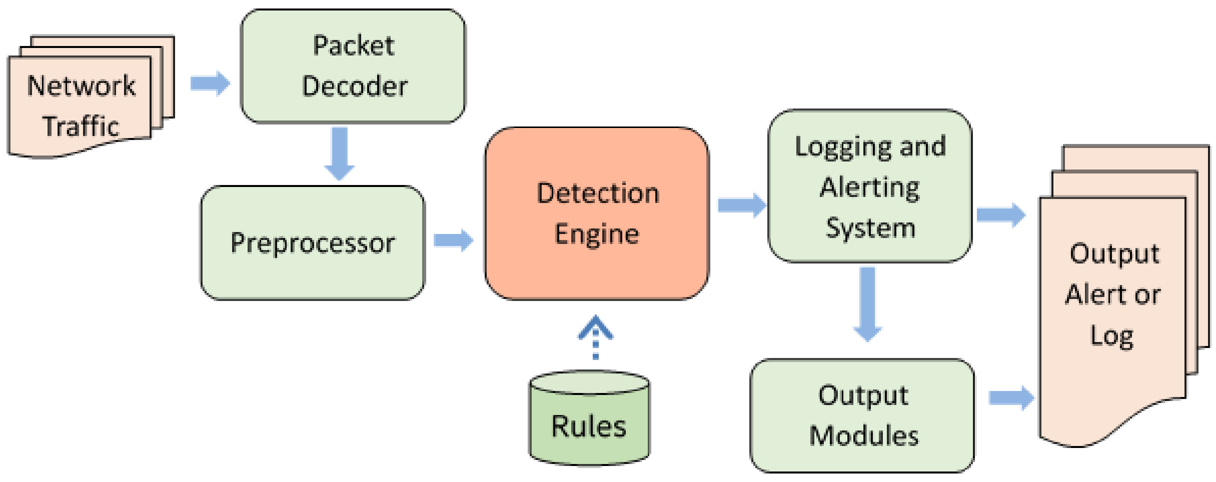

Illustration 1: The Architecture of Snort (from software.intel.com)

We can distinguish three major components, the preprocessor, the detection engine and the output mechanism. All of them are plugins.

- The preprocessor aims to prepare the received packet for the detection engine. It can for example detect anomalies in the headers, perform packet defragmentation and decode HTTP URI.
- The detection engine will apply rules to packets. If a rule matches the data in the packet, the packet information is then sent to the output mechanism.
- The output mechanism will process alerts/logs and generate the final output. These alerts can be logged into a file, a UNIX socket, stored in a database, sent to syslog, … User interfaces are available such as Snorby, SQueRT or BASE.

The plugins system allows packet inspection and manipulation at multiple stages. Plugins can have effects before packets are being handled by the detection engine, only perform a simple tests on the packet or report results from other plugins.

Snort placement in the network is crucial. If it is misplaced, packets may not be seen. Ideally Snort can be deployed in different locations because of its lightweight design. The network interface that will sniff incoming packets must be set to promiscuous mode so that all traffic can be seen. Once Snort has been compiled/installed we can adjust the configuration if necessary (NIDS mode) and run it with command line parameters.

2. Detection Rules

Detection rules are at the core of Snort detection mechanism, we will describe their structure and how they are used by the detection engine.

Snort rules form what are called “signatures”. A rule will describe a pattern, modular detection elements are used to do so. This rule system is very flexible and allows a wide range of detection capabilities (OS fingerprinting, buffer overflows, stealth scans, ...). The rules are specified in the Snort configuration file. Example rules are provided with the software (dos.rules, virus.rules, backdoor.rules, scan.rules, ...).

The structure of a snort rule goes as follows:

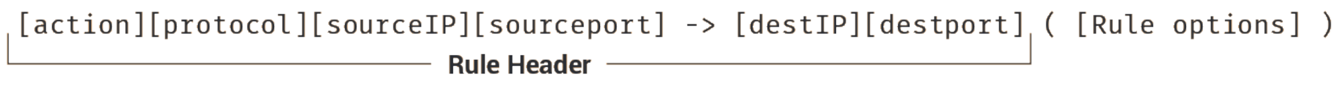
[action][protocol][sourceIP][sourceport] -> [destIP][destport] ([Rule options])
Rule Header

Illustration 2: Basic outline of a Snort rule (from snort.org)

A rule is composed of two parts, The Header and the Options. The first item in a rule header is the action. It specifies the action Snort will take if a packet matches the rule. Then comes the protocol at the infrastructure layer (Transport + Network) like TCP, UDP or IP. The source IP address/range and port of the incoming packet are specified, they can be set to the value "any". The direction operator is represented by an arrow, it indicates the orientation of the traffic to which the rule applies. Destination IP address/range and port are at the end of the rule header.

Options can be added to rules. They are separated by a semicolon and the option keywords are separated from their arguments with a colon. Options can contains elements such as a specific message to display (msg: "Suspicious activity detected"), a rule identifier (sid:1022), a revision number (rev:42), a specific content (content:"backdoor"), a Perl-compatible regular expression (pcre).

The detection engine assembles all the rules in multiple trees. The top of a tree is built from the information found in the rule header, such as the action, the protocol and the hosts endpoints. Linked to this branch, the rule options are added similarly as shown here:

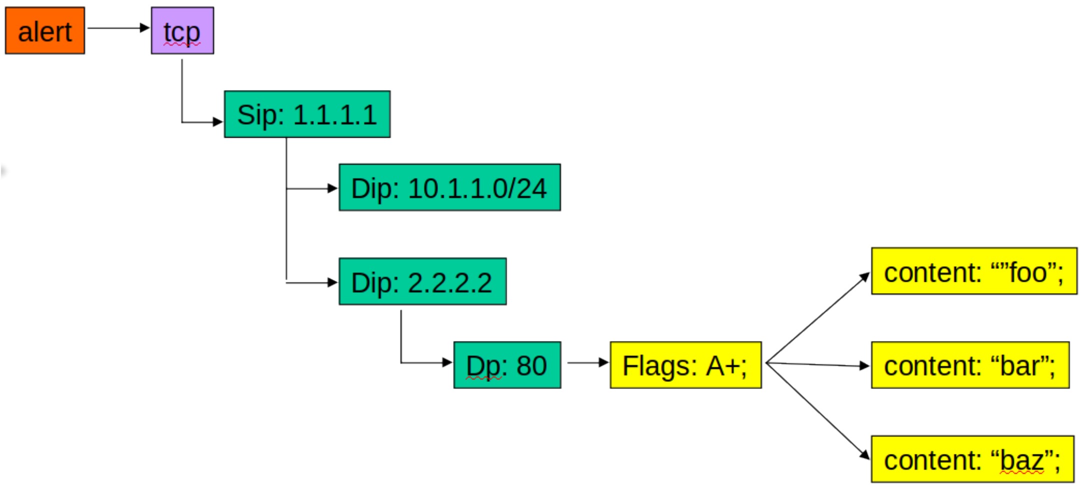

Illustration 3: Snort detection engine example tree [2]

Snort detection engine evaluates rules in this specific order: alert, pass and log. A priority is also assigned according to the matching complexity (IP header, TCP header, Application protocol header, content).

The pattern matching is based on the Boyer/Moore string search algorithm. Variants such as the Aho–Corasick algorithm and Boyer-Moore-Horspool were proposed and integrated.

3. PCRE Matching

PCRE (Perl Compatible Regular Expressions) matching brings flexibility in the detection rules. In this section we will outline their importance.

Snort allows regular expression pattern matching using almost the same syntax and semantics as in the Perl programming language. This feature is specified in the rule options. Here is the format:

```
pcre:[!]"(/<regex>/|m<delim><regex><delim>)[ismxAEGRUBPHMCOIDKYS]";
```

An example could be to match a module code we had this year in the MSc Cyber Security (Ex. CO892):

```
pcre:"/(CO|EL)8[0-9]{2}/";
```

The usage of regular expressions in content matching is of great help as the name of files, IP addresses and SQL queries will differ but keep the same pattern. This enables us to match content in a generic and flexible manner. However, their usage is computationally expensive.

4. Example rules

Several examples of rules will be provided within this section.

1. This rule will be triggered if the word “SECURITY” is found:

   ```
   alert tcp any any → any any (content: “SECURITY”; msg: “SECURITY keyword
   found”;)
   ```

2. This rule will be triggered if the space insensitive “Hello world” phrase is found:

   ```
   alert tcp any any → any any (pcre: ”/Hello\s+world/”; msg: “Hello world!”;)
   ```

3. This rule will be triggered if a single word of text enclosed in double quotes that starts with a capital letter and is between four and seven letters long is sent to a mail server:

   ```
   alert tcp any any → any 25 (pcre: ”/\”[A-Z]+[a-zA-Z]{3,6}\”/”; msg:
   “Interesting word”)
   ```

4. This rule will be triggered if a UK national insurance number is sent to a web server:

   ```
   alert tcp any any → any 80 (pcre: “/[A-CEGHJ-NOPR-TW-Z]{2}(?:\s*\d\s*){6}
   [ABCD\s]{1}/”; msg: “UK national insurance number detected”)
   ```

5. References

- [1] Snort official website
- [2] Snort presentation by the initial author Martin Roesch
- [3] Signature-based presentation by Pavel Laskow