

The Snort Intrusion Detection System

🕒 9 minute read

This post is an overview of the Snort IDS/IPS. Details are given about its modes, components, and example rules. I originally wrote this report while pursuing my MSc in Computer Security.

Snort Overview

Snort is an open source *Network Intrusion Detection System* [1] (NIDS). NIDS are responsible for analyzing traffic from a network, and testing each packet against a list of rules. If a packet corresponds to a rule, the NIDS can log the event, send an alert, and/or take an action such as dropping the packet. We will first take a look at what Snort does, and how it works. We will then examine how rules used by the detection engine are formed, and what role *Pearl Compatible Regular Expression (PCRE)* has in these rules. Finally we will finish with examples of rules, particularly rules that demonstrate the importance of PCRE.

Snort Modes

Snort is a Network Intrusion Detection System, but comes with three modes of operation, all of which are parts of the NIDS in itself. The first mode, **Sniffer Mode** [2], displays packets that transit over the network. It may be configured to display various types of packets (TCP, UDP, ICMP), as well as what to display of the packets themselves, either the headers or packet data as well.

The second mode of operation granted by snort is the **Packet Logger Mode** [3]. It allows the user to save packets detected from Sniffer Mode to be saved to the hard disk. Through this mode, the user may specify rules indicating which packets to save, for example, to save only packets relative to (going to, or coming from) a specific address.

Finally, the last mode is the **NIDS Mode** [4]. This mode is very similar to the packet logger, but allows for more specific rules to be applied to packets, refining the packets that are in fact logged (or alerted). The rules applied are specified, or included in the configuration file that is passed as a parameter when launching snort.

It must be noted that each of these modes have various options which may be configured through command line parameters or even configuration files. For example, with alerts triggered in NIDS mode, you may configure the contents of the alerts, where the alerts are stored, or even if you just want to output to console, or through a UNIX socket to another program. While the input for the

NIDS is typically done with the Sniffer Mode, this can be replaced with a .pcap (packet capture) file if sniffing is not an option; either the intrusion already occurred, or you cannot place Snort in a location to allow sniffing.

Snort Components

Snort is composed of four major components, which chained together allows snort to fulfill its various modes. The first component is the **Decoder** which is responsible for forming packets to be used by the other components. As mentioned in the Snort FAQ, it *“has the job of determining which underlying protocols are used in the packet”* [5] as well as determining the location and size of the packet data which is then used in later components. It should be noted that the Decoder also looks for anomalies in headers (such as invalid sizes), which may then cause it to generate alerts.

The next major components are the **Preprocessors**. These components work as plugins, and are able to arrange or modify packet data. This allows services (such as HTTP [6], or FTP [7]) to have a corresponding preprocessor to verify anomalies specific to that service. Its job is ultimately to try and make it harder to fool the detection engine. Examples of how it can do this are by decoding URI's, defragmenting packets [8] (Fragmentation of packets can be used to fool the detection engine), detect port scanning [9], as well as detect anomalies in ARP packets [10], such as ARP spoofing.

The primary component, the Detection Engine has the responsibility to *“detect if any intrusion activity exists in a packet”* [11]. It does this by chaining together sets of rules, specified in configuration files which include these rules, and applying them to each packet. If the packet matches a rule, the specified action of that rule is taken, or the packet is dropped. If snort is performing this in real time, depending on the network load, latency may be experienced, with worst case scenarios resulting in packets being dropped all together.

If a packet is matched to a rule, the log and or alert will be generated by the **Alert and Logging System**. The message and contents generated by this component can of course be configured through the configuration file. If a packet triggers multiple rules, the highest alert level is what will actually be generated by this component.

Finally, after an alert or log is generated, it passes through the **Output Modules** component. This component is tasked with controlling the type of output generated, uses a plugin system [12] giving the user flexibility, and is also highly configurable. This may include simply logging, or logging to a database, sending SNMP traps, generating XML reports, or even sending alerts through UNIX sockets, allowing for (for example) dynamic modification of network configurations (Firewalls or Routers).

Snort Rules

As previously mentioned, rules are used throughout components to detect anomalies in packets. Rules may be applied to Network and Transport Layer headers (IP, TCP, UDP, ICMP), or even Application layer headers (FTP, HTTP, etc.), but of course the rules can also be applied to packet data (the payload).

Rules are composed of two parts, a **Rule Header**, which specifies what action should be taken in case of a match, the type of packet (TCP, UDP, etc.), as well as source and destination IP's and port numbers. The last part is the **Rule Options**, which specifies content that flags packets as a match, the overall rule will take the following form:

```
action protocol source port -> destination port (options)
```

</>

It should be noted that while most options are optional, the *sid* (Snort ID) is required, and should not conflict with the SID of another rule. It is the unique identifier given to each rule. Snort reserves SIDs from 0 - 1,000,000. [13]

In the rule options, amongst a long list of possible flags that may be used to detect various bits of data in packets, users may include **Pearl Compatible Regular Expressions** through the option *pcr*. This allows the detection of data in the packet by using Regular Expressions, giving rules more control and flexibility. PCRE takes the standard format `/expression/flags`, although double quotes, semi-colons, and forward-slashes must be escaped.

Rule Examples

Security

Alert if the packet contains the word SECURITY.

Rule

```
alert ip any any -> any any (sid:1000001;msg:"Word SECURITY found";content:"SECURITY";)
```

</>

Description

In this example, we can notice a few things:

- `alert` : this allows us to trigger an alert if rule matches
- `ip` : this allows the rules to be matched against any protocol (TCP, UDP, or ICMP)
- `any any -> any any` : any source host and port to any destination host and port
- `sid:1000001;msg:"Word SECURITY found"` : the ID of the rule, and the message to send with the alert.

The particularity of this rule is the option content. As the Snort manual describes *"Whenever a content option pattern match is performed, the Boyer-Moore pattern match function is called and the (rather computationally expensive) test is performed against the packet contents"* [14]. As we can tell from this, here I'm simply looking for `SECURITY`, in a case sensitive match.

Hello World

Alert if the packet contains the phrase "Hello World", with one or more spaces between Hello and World.

Rule

```
alert ip any any -> any any (sid:1000002;msg:"Phrase Hello World found";pcre:"/Hello\s+World/");
```

Description

The only new feature in this example is the presence of *pcre* instead of *content* as an option. As the name suggest, it allows us to use Pearl Compatible Regular Expressions. This regular expression looks for Hello followed by one or more spaces (`\s+`) followed by World.

Mail Server

Alert if a packet from any computer to a mail server contains a single word of text enclosed in double quotes, which starts with a capital letter, and is between four and seven letters long.

Rule

```
alert ip any any -> any smtp (sid:1000002;msg:"Double quoted text sent to mail server found";pcre:"/[A-Z][a-zA-Z]{3,6}\"/");
```

Description

Here we use the destination port to specify packets going to mail servers (smtp can be replaced by 25). Backslashes are required in the PCRE option to escape quotes. The PCRE used looks for any string starting with a double quote, followed by a capital (`[A-Z]`), with a total of 4-7 letters in length, thus the capital plus 3 to 6 letters (`[a-zA-Z]{3,6}`), followed by a double quote.

Web Server

Alert if a packet from any computer is sent to a web server containing a UK national insurance number.

Rule

```
alert ip any any -> any http (sid:1000003;msg"UK national insurance number found";pcre:"/([A-CEGHJ-PR-TW-Z]
[A-CEGHJ-NPR-TW-Z])(\s*[0-9]){6}([A-D]|\s)/i";)
```

Description

Alright, this one is a little more complicated and demonstrates the use of sets in PCRE. It is based on the specifications of NIM391110 which may be found here:

<http://www.hmrc.gov.uk/manuals/nimmanual/nim39110.htm>

(<http://www.hmrc.gov.uk/manuals/nimmanual/nim39110.htm>):

- `([A-CEGHJ-PR-TW-Z][A-CEGHJ-NPR-TW-Z])` : First capture group, this searches for a two character pair. The first and second characters cannot be D, F, I, Q, U, or V. The second character has the added restraint of not being O.
- `(\s*[0-9]){6}` : Second capture group, searches for a number between 0 and 9, repeated 6 times, allowing for zero to multiple spacing (`\s*`). It is important not to add a `\s*` at the end of this rule, because of the next group constraint being either a space or a letter, multiple spaces after could thus lead to errors in matching.
- `([A-D]|\s)` : A character between A and D or a space.
- `i` : Flag insensitive, allows for case insensitive matching (thus capital or lowercase characters for matching)

References

[1] S. T. /. O. S. Community, "What is Snort?," Cisco Systems, [Online]. Available: <https://snort.org/faq/what-is-snort>.

[2] S. Team, "1.2 Sniffer Mode," Cisco Systems, [Online]. Available: <http://manual.snort.org/node4.html>.

[3] S. Team, "1.3 Packet Logger Mode," Cisco Systems, [Online]. Available: <http://manual.snort.org/node5.html>.

[4] S. Team, "1.4 Network Intrusion Detection System Mode," Cisco Systems, [Online]. Available: <http://manual.snort.org/node6.html>.

[5] S. T. /. O. S. Community, "README.decode," Cisco Systems, [Online]. Available: <https://www.snort.org/faq/readme-decode>.

[6] S. Team, "2.2.7 HTTP Inspect," Cisco Systems, [Online]. Available: <http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node17.html#SECTION00327000000000000000>.

[7] S. Team, "2.2.11 FTP/Telnet Preprocessor," Cisco Systems, [Online]. Available: <http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node17.html#SECTION0032110000000000000000>.

[8] S. Team, "2.2.1 Frag3," Cisco Systems, [Online]. Available: <http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node17.html#SECTION0032100000000000000000>.

[9] S. Team, "2.2.4 sfPortscan," Cisco Systems, [Online]. Available: <http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node17.html#SECTION0032400000000000000000>.

[10] S. Team, "2.2.15 ARP Spoof Preprocessor," Cisco Systems, [Online]. Available: <http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node17.html#SECTION0032150000000000000000>.

[11] R. U. Rehman, "Intrusion Detection," in Intrusion Detection With SNORT, Apache, MySQL, PHP, And ACID: Advanced IDS Techniques Using SNORT, Apache, MySQL, PHP and ACID, Pearson Technology Group, 2007, p. 14.

[12] S. Team, "2.6 Output Modules," Cisco Systems, [Online]. Available: <http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node21.html>.

[13] S. Team, "3.4.4 sid," Cisco Systems, [Online]. Available: <http://manual.snort.org/node31.html#SECTION0044400000000000000000>.

[14] S. Team, "content," Cisco Systems, [Online]. Available: <http://manual.snort.org/node328.html>.

 **Tags:**


IDS/IPS (<https://kevinalmansa.github.io/tags/#ids-ips>)

Monitoring (<https://kevinalmansa.github.io/tags/#monitoring>)

Snort (<https://kevinalmansa.github.io/tags/#snort>)

 **Categories:**

IDS/IPS (<https://kevinalmansa.github.io/categories/#ids-ips>)

 **Updated:** July 08, 2017

LEAVE A COMMENT

0 Comments

 **1** **Login** ▼

G

Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name

 **2**

Share

Best **Newest** **Oldest**

Be the first to comment.

Subscribe

Privacy

Do Not Sell My Data