First Impressions: Completely revised Snort Version 2.0

Super Sniffer

The Snort Network Intrusion Detection System is the most commonly used NIDS worldwide. Version 2.0, which is due for release shortly, recognizes attacks up to 18 times quicker than its predecessor. BY RALF SPENNEBERG

Snort, the Open Source NIDS (Network Intrusion Detection System) by Martin Roesch, has developed into a powerful tool since it was first released in 1998. Originally a simple packet sniffer based on the Libpcap library, the tool simply compared packets to a very simple signature database.

By now Snort has moved up the scale from a lightweight NIDS to an all encompassing system that performs well when compared with expensive, commercial products. Amongst other features Snort offers IP defragmenting, TCP stream reassembly and UNICODE decoding.

These terms are indicative of the kind of packet normalization that Snort performs to provide for uniform data representation.

After this step NIDS compares the packets with rules stored in an comprehensive signature database, thus recognizing potential attacks. When Snort detects an attack, it uses a number of output plug-ins to alert whoever is responsible.

This article introduces the enhancements found in the Snort 2.0 release candidate, and describes their configuration, avoiding any functions already available in previous versions. Refer to the article at [2] for more information.

THE AUTHOR

Ralf Spenneberg is a freelance Unix/Linux trainer and author. Last year saw the release of his first book: "Intrusion Detection Systems for Linux Servers". Ralf has also developed various training materials.

What's New in Snort 2.0?

In fact, Snort has a whole bunch of new features to offer. Some of them will be immediately apparent to administrators. For example, a revised port scan detector, *portscan2*, a detector for polymorphic shell code *fnord*, and a performance monitor, *perfmonitor*, have been available since Version 1.9. Version 2.0 adds a new pre-processor HTTP Flow. Unfortunately, Fnord did not make it into Version 2.0.

However, the major enhancements in Version 2.0 mainly apply to Snort itself. The developers have completely replaced the detection engine with a new and quicker version.

HTTP Flow

As a pre-processor, the HTTP Flow Analyzer is capable of splitting communications between a webclient and a webserver into two parts. The detection engine can handle these flows separately. Snort assumes that about five percent of the traffic will be caused by the client and the remaining 95 percent by the server. The headers in a server response are only a few hundred bytes, and comprise a mere three to five percent of data transfers.

The HTTP Flow Analyzer makes use of this knowledge and offloads only the client flow and the headers from the server flows to the detection engine, as malevolent attacks should be apparent based on this information. This typically helps reduce the data that needs analyzing by 80 percent.

The pre-processor supports two different modes; Quick and Full. In Quick mode, it individually examines each packet.

To do so, it needs to know on what ports the webserver is listening. Also, it needs to know the number of bytes to be investigated in the webserver response. A typical call is as follows:

```
preprocessor httpflow: quick ➥
  depth 200 ports 80 3128
```

This will cause the pre-processor to analyze the first 200 bytes of the server responses originating at ports 80 and 3128.

In Full mode HTTP Flow uses the Stream4 pre-processor to reassemble the individual packets that make up the traffic. Stream4 only releases packets with a valid TCP status for analysis. In this mode, you do not need to supply the header lengths, as the pre-processor will automatically recognize them.

New Addon: Snort 2.0 Perl Patch

Brian Caswell and Jeff Nathan presented a new Perl Patch in their CanSecWest/core03 presentation: Advanced IDS [5]. This new Snort Perl Patch for Snort 2.0 adds two new keywords to the Snort rule language: *perl* and *perlre*.

The *perlre* keyword will perform a full regular expression matching on the contents of the inspected packet. The *perl* keyword provides runtime execution of any perl code. The perl code is stored in a separate file *snort.pl* which allows for an easy update and modification by the user.

Using these keywords much more complex rules can be created.

```
preprocessor httpflow: full
```

As Snort applies pre-processors in the order stipulated in the configuration file, HTTP Flow must occur after Frag2 and Stream4 in the configuration file.

Portscan2 and Perfmonitor

The new port scan detector, Portscan2, uses its own state table to recognize more scans with less errors. Four directives are available for configuring the detector; see Listing 1.

These parameters allow the detector to recognize port scans that either contact five different systems within 60 seconds, or talk to 20 different ports. As the detector logs the connections, it should be able to distinguish a SYN/ACK scan from a genuine connection. The heritage port scan detector was only capable of detecting scans that opened at least four ports in less than three seconds.

The Perfmonitor provides current statistics on Snort's activities, including statistics on the network flows, and any events Snort has detected. Perfmonitor needs an update parameter for the events that can either be supplied on a (*time*) basis or as a packet count (*pktcnt*).

Admins can use Perfmonitor to monitor Snort's current performance. The output (Listing 2 provides an excerpt) also provide far more detail than the statistical output that Snort typically provides when terminating.

Rule Optimization

The Rule Optimizer and the High Performance Multirule Inspection Engine are responsible for Snort 2.0's improved speed. Older versions checked the parameters defined in the rules individually and in sequence. The test stopped when a match had been found or all rules had been processed. Enter the new Rule Optimizer. It creates subgroups of rules, which it sorts by specific criteria, such as the source or target port.

Now, when Snort wants to analyze a packet, it simply works its way through the appropriate subgroup. Within the groups there is an additional subdivision between rules that test the content of a packet (*content*, *uricontent*), and rules that simply inspect the header.

The Multirule Inspection Engine uses the Wu-Manber algorithm to perform extremely quick parallel searches for specified strings within the packet content. If the algorithm finds a match, Snort goes on to check the other parameters for the rule. In contrast to earlier versions, the NIDS first checks the packet content, and then the header. If the whole rule applies, Snort adds the event to a queue.

The other rules are applied after the content rules, and Snort 2.0 follows the traditional pattern when working its way through them. After working its way through all the rules, Snort parses the queue and selects an event for logging, where Uricontent has priority over Content and both have priority over normal rules. This ensures that the log will always contain the entry that most precisely describes the event.

Conclusion

The development work put into optimizing Snort 2.0's rule-sets really shows. A growing number of rules inspect packet content for specific byte or character strings. As somewhere in the region of 70 to 90 percent of all network traffic is caused by HTTP connections, the HTTP Flow pre-processor is particularly effective, drastically reducing the amount of data and thus the detection overhead.

The classical rule parsing scheme provided by older Snort versions soon reached its limits with rule-sets comprising 1,500 entries.

In combination with some less significant advancements in the Snort rule language, and the HTTP Flow preprocessor allows Snort 2.0 to run on a gigabit speed network without dropping packets, despite more complex rule-sets. ■

Listing 1: Port Scan Detector

```
preprocessor portscan2-ignorehosts: $HOME_NET
preprocessor portscan2-ignoreports-from: 53 80
preprocessor portscan2-ignoreports-to: 53 80
preprocessor portscan2: scanners_max 3200, \
  targets_max 5000, target_limit 5, \
  port_limit 20, timeout 60, log
```

Listing 2: Perfmonitor Output

```
Snort Realtime Performance
(Mon Apr 7 14:15:56 2003)
--------------------------
Pkts Recv:  9991
Pkts Drop:  0
% Dropped:  0.00%

KPkts/Sec:  0.02
Bytes/Pkt:  723

Mbits/Sec:  0.11 (wire)
Mbits/Sec:  0.00 (rebuilt)
Mbits/Sec:  0.11 (total)

PatMatch:   87.50%

CPU Usage:  0.11% (user) 0.03% ➥
(sys) 99.86% (idle)

Alerts/Sec      :  0.1
Syns/Sec        :  0.1
[...]

Protocol Byte Flows - %Total Flow
---------------------------------
TCP:   99.85%
UDP:   0.11%
ICMP:  0.00%
OTHER: 0.04%
[...]

TCP Port Flows
--------------
Port[80] 0.42% of Total, Src: ➥
79.46% Dst:  20.54%
Port[993] 1.13% of Total, Src: ➥
77.03% Dst:  22.97%
Ports[High<->High]: 98.36%
[...]
```

INFO

[1] Snort: *http://www.snort.org*

[2] Ralf Hildebrand, "Cain and Abel, Snort and Nmap – two sides of the same coin", Linux Magazine, Issue 4, p46 , or online archive *http://www.linux-magazine.com/issue/04/snort_nmap.pdf*

[3] Whitepaper on Snort 2.0: *http://www.snort.org/docs/*

[4] Snort 2.0 RPM packages: *http://www.spenneberg.org/IDS*

[5] Presentation Advanced IDS and Snort Perl Patch: *http://cerberus.sourcefire.com/~jeff/presentations/cansecwest-2003/*