

Super-Schnüffler

Snort ist das weltweit am häufigsten eingesetzte Network Intrusion Detection System. Die kurz vor der Veröffentlichung stehende Version 2.0 ist beim Erkennen von Angriffen bis zu 18-mal schneller als ihre Vorgängerin. Sogar in Gigabit-Netzwerken erschnüffelt Snort jedes faule Datenpaket. Ralf Spenneberg



Snort, das Open-Source-NIDS (Network Intrusion Detection System) von Martin Roesch, hat sich seit seiner ersten Veröffentlichung im Jahre 1998 zu einem mächtigen Werkzeug entwickelt. Ursprünglich handelte es sich um einen einfachen Packet Sniffer auf Basis der Libpcap-Bibliothek, der die Pakete mit einer sehr einfachen Signaturdatenbank verglich. Inzwischen ist Snort von einem Leichtgewicht-NIDS zu einem recht umfangreichen System angewachsen und braucht den Vergleich mit kommerziellen und hochpreisigen Lösungen nicht zu scheuen.

Snort verfügt unter anderem über IP-Defragmentierung, TCP-Stream-Reassemblierung und Unicode-Dekodierung. Hinter diesen Begriffen steckt die Normalisierung der Pakete, bei der Snort für eine einheitliche Darstellung der Daten sorgt. Danach vergleicht das NIDS die Pakete mit den Regeln seiner umfangreichen Signaturdatenbank und erkennt so mögliche Angriffe. Hat es eine Attacke be-

merkt, meldet es den Vorfall über eines seiner vielen Ausgabe-Plugins.

Dieser Artikel stellt die wesentlichen Neuerungen des Release-Kandidaten von Snort 2.0 vor und erläutert die Konfiguration. Funktionen, die in den vorherigen Versionen bereits enthalten waren, sind dabei bewusst ausgeklammert. Mehr dazu ist in verschiedenen Artikeln [3] und Büchern [2] zu finden.

Neu in Snort 2.0

Die Version 2.0 weist eine ganze Reihe von Neuerungen auf, einige sind für den Administrator offen erkennbar. So gibt es seit Snort 1.9 einen überarbeiteten Portscan-Detektor »portscan2«, einen Detektor für polymorphen Shellcode »fnord« sowie einen Performance-Monitor »perfmomitor«. Die Version 2.0 bringt zudem den neuen Präprozessor HTTP-Flow mit. Fnord wurde leider noch nicht in die Version 2.0 übernommen.

Die wichtigste Neuerung der Version 2.0 betrifft aber Snort selbst. Die Entwickler haben die gesamte Detektionsmaschine ausgetauscht und durch eine neue, wesentlich schnellere Variante ersetzt.

HTTP-Flow

Der HTTP-Flow-Analysator ist als Präprozessor in der Lage, die Kommunikation zwischen einem Webclient und einem Webserver in zwei Teile aufzuteilen, den Client-Flow und den Server-Flow. Die Detektionsmaschine behandelt dann diese Flüsse getrennt.

Snort geht davon aus, dass bei einer HTTP-Kommunikation etwa fünf Prozent des Verkehrs auf den Client und 95 Prozent auf den Server entfallen. Bei der

Antwort des Servers sind die Header nur einige hundert Byte lang. Meist umfassen sie nur drei bis fünf Prozent der übertragenen Daten. Der HTTP-Flow-Analysator nutzt dieses Wissen und belastet die Detektionsmaschine nur mit dem Client-Flow und den Headern des Server-Flows, da sich hier bereits die bösartigen Angriffe offenbaren. Das reduziert die zu analysierenden Daten typischerweise um 80 Prozent.

Dieser Präprozessor unterstützt zwei unterschiedliche Modi: Quick und Full. Im Quick-Modus untersucht er jedes Paket einzeln. Dazu muss er wissen, auf welchen Ports die Webserver lauschen. Zusätzlich benötigt er die Anzahl der zu untersuchenden Bytes in der Antwort des Webserver. Ein typischer Aufruf lautet:

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

Der Präprozessor analysiert die ersten 200 Bytes der Server-Antworten, die von den Ports 80 und 3128 stammen.

Im Full-Modus benutzt HTTP-Flow den Stream4-Präprozessor, um die Einzelpakete der Kommunikation wieder zusammensetzen. Stream4 gibt nur Pakete zur Analyse weiter, deren TCP-Zustand er validiert hat. In diesem Modus ist es nicht nötig, die Header-Länge anzugeben. Der Präprozessor erkennt sie selbstständig.

```
preprocessor httpflow: full
```

Snort wendet die Präprozessoren in der Reihenfolge an, in der sie in der Konfigurationsdatei aufeinander folgen. Daher ist es im Full-Modus zwingend erforderlich, dass HTTP-Flow erst nach Frag2 und Stream4 steht.

Der neue Portscan-Detektor Portscan2 benutzt eine eigene Zustandstabelle, um Scans besser zu erkennen und dabei weniger falsche Alarme auszulösen. Der Detektor lässt sich über vier Direktiven konfigurieren (siehe [Listing 1](#)).

Portscan2 und Perfmonitor

Mit diesen Parametern erkennt er Portscans, die innerhalb von 60 Sekunden entweder fünf verschiedene Systeme kontaktieren oder 20 verschiedene Ports ansprechen. Da er über die Verbindungen Buch führt, soll er in der Lage sein, einen SYN/ACK-Scan von einer echten Verbindung zu unterscheiden. Der klassische Portscan-Detektor war nur in der Lage, Scans zu erkennen, die Verbindungen zu mindestens vier Ports in weniger als drei Sekunden öffnen.

Der Perfmonitor gibt aktuelle Zahlen über die Arbeit von Snort aus. Damit lassen sich Statistiken über die Netzwerkflüsse und die von Snort erkannten Ereignisse führen. Perfmonitor benötigt die Angabe eines Zeitraums, nach dem er die Ergebnisse aktualisieren soll. Diese Konfiguration erfolgt entweder als Zeiteinheit (»time«) oder als Zahl von Netzwerkpaketen (»pktcnt«).

Mit Perfmonitor kann der Admin die Leistung von Snort fortlaufend überwachen. Die Ausgaben (ein Auszug ist in [Listing 2](#) zu sehen) sind zudem wesentlich detaillierter als die statistischen Informationen, die Snort beim Beenden üblicherweise anzeigt.

Regeloptimierung

Der Rule Optimizer und die High Performance Multirule Inspection Engine sind für die Geschwindigkeitssteigerung von Snort 2.0 verantwortlich. Ältere Versionen prüften die in den Regeln definierten Parameter einzeln nacheinander. Der

Test endete, wenn eine Regel zutraf oder alle Regeln abgearbeitet waren. Der neue Rule Optimizer greift hier ein. Er erzeugt aus den Regelsätzen Untergruppen, die er nach bestimmten Kriterien wie Quell- und Zielport einteilt.

Wenn nun Snort ein Paket analysiert, muss es nur die zutreffende Untergruppe abarbeiten. In den Gruppen wird unterschieden zwischen Regeln, die auch den Inhalt des Pakets (»content«, »uricontent«) testen, und Regeln, die lediglich den Header betrachten.

Die Multirule Inspection Engine sucht mit dem Wu-Manber-Algorithmus parallel und sehr schnell nach sämtlichen definierten Zeichenfolgen im Paketinhalt. Findet der Algorithmus eine Übereinstimmung, prüft Snort die anderen Parameter der entsprechenden Regel. Im Gegensatz zu früher prüft das NIDS also zunächst den Paketinhalt und dann erst den Header. Trifft die gesamte Regel zu, trägt Snort dies als ein Ereignis in einer Warteschlange ein.

Nach den Content-Regeln folgen die restlichen Regeln, die Snort 2.0 wie in den früheren Versionen behandelt. Nachdem es alle Regeln untersucht hat, arbeitet Snort die Warteschlange ab und wählt ein Ereignis zur Protokollierung aus: Bevorzugt nimmt es Uricontent vor Content und beide vor den normalen Regeln. So finden sich nur die Treffer im Log, die den Vorfall am genauesten beschreiben.

Fazit

Die Optimierung in Snort 2.0 trägt der Entwicklung der Regelsätze Rechnung. Immer mehr Regeln überprüfen den Paketinhalt auf bestimmte Byte- oder Zeichenfolgen. Da etwa 70 bis 90 Prozent

des Netzwerkverkehrs aus HTTP-Verbindungen bestehen, wirkt sich der HTTP-Flow-Präprozessor besonders günstig aus. Er reduziert die zu prüfende Datenmenge und senkt damit den Detektionsaufwand erheblich.

Die klassische Regelprüfung älterer Snort-Versionen kam bei Regelsätzen mit 1500 Einträgen schnell an ihre Grenzen. Kombiniert mit einigen kleineren Verbesserungen der Snort-Regelsprache und dem HTTP-Flow-Präprozessor ist Snort 2.0 in der Lage, in einem Gigabit-Netzwerk ohne Paketverlust zu arbeiten. Dies gelingt ihm auch bei umfangreichen Regelsätzen. (fjl) ■

Listing 1: Portscan-Detektor

```
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-Ausgabe

```
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%
[...]
```

Der Autor

Ralf Spenneberg arbeitet als freier Unix/Linux-



Trainer und Autor. Er veröffentlichte letztes Jahr sein erstes Buch „Intrusion Detection für Linux-Server“ und entwickelte bereits mehrere Kursunterlagen.

Infos

- [1] Snort: [\[http://www.snort.org\]](http://www.snort.org)
- [2] Ralf Spenneberg, „Intrusion Detection für Linux-Server“: Markt und Technik Verlag, 2003
- [3] Ralf Hildebrand, „Kain und Abel - Snort und Nmap, zwei Seiten derselben Münze“: Linux-Magazin 12/00, S. 102
- [4] Whitepaper zu Snort 2.0: [\[http://www.snort.org/docs/\]](http://www.snort.org/docs/)
- [5] Snort 2.0rc2, RPM-Pakete: [\[http://www.spenneberg.org/IDS\]](http://www.spenneberg.org/IDS)