

Hogwash, ein Gateway-IDS auf der Basis von Snort

Netzwerk-Schrubber

Hogwash kombiniert die Funktion eines Intrusion-Detection-Systems (IDS) mit den Aufgaben einer Firewall. Snort-ähnliche Regeln entscheiden, ob das Gateway ein Paket weiterleitet oder sperrt. Ähnlich einer Bridge arbeitet Hogwash als transparente Firewall ohne eigene IP-Adressen. Ralf Spenneberg



Firewalling und IDS (Intrusion Detection System) sind eigentlich zwei gut abgegrenzte Bereiche: Ersteres lässt nur bestimmte Pakete und Verbindungen zu, Letzteres erkennt Angriffsversuche und informiert die Admins. Da ein IDS nur bekannte Angriffe bemerkt, sollte der Admin die jeweilige Sicherheitslücke bereits geschlossen haben. Dennoch gibt es Situationen, in denen das nicht mög-

lich ist: Hier helfen IDS-Varianten, die beim Erkennen eines Angriffs ihn auch gleich unterbinden.

Diese Funktion ist in Hogwash [1] implementiert. Wie es zu diesem Gateway-IDS kam, fasst der **Kasten „Historie“** zusammen – im Folgenden stehen Installation, Konfiguration und Einsatz von Hogwash im Mittelpunkt.

Vielseitiger Paketschrubber

Hogwash unterstützt drei verschiedene Modi: IDS, Scrubber und Bait'n'Switch. Im IDS-Modus überwacht Hogwash den Netzwerkverkehr und alarmiert bei bössartigen Paketen die Admins. Es ist dabei zwar nicht in der Lage, die Pakete abzufangen, kann die Verbindungen aber mit gefälschten TCP-Resets abbrechen. Bei entsprechender Konfiguration überwacht Hogwash den Verkehr auf mehreren Schnittstellen.

Im Paketschrubber-Modus (Scrub) filtert Hogwash die Pakete. Es fälscht dafür TCP-Resets, verwirft die Pakete oder modifiziert sie und vereitelt damit wirkungsvoll Angriffe. Hogwash nutzt bis zu 16 Netzwerkkarten und leitet bei Be-

darf die Pakete zwischen diesen Devices weiter – allerdings nicht als Router, sondern als Bridge. Die Software arbeitet transparent auf dem Link-Layer im Promiscuous-Modus. Der Admin kann sogar den IP-Stack des Betriebssystems deaktivieren. Ein IP-Forwarding muss er sogar abschalten, da Hogwash nun das Weiterleiten übernimmt.

Im experimentellen Bait'n'Switch-Modus schützt Hogwash ein Produktionssystem, vereitelt aber keine Angriffe. Vielmehr leitet es die suspekten Verbindungen auf einen Honeypot zur genaueren Analyse um: Ein Einbruch auf dieser Maschine hat keine negativen Folgen.

Installation

Hogwash lässt sich recht einfach installieren. Nach dem Herunterladen und Auspacken folgt der übliche Configure-Make-Aufruf:

```
./configure
make
```

Hogwash arbeitet auf jeder beliebigen Linux-Distribution. Es bietet sich jedoch an, den IP-Stack des Betriebssystems zu

Listing 1: Konfigurationsdatei

```
01 <system>
02 Name=Hogwash IDS Gateway
03 ID=1001
04 Threads=1
05 </system>
06
07 <interface eth0>
08 Type=linux_raw
09 Proto=Ethernet
10 Role=External
11 </interface>
12
13 <interface eth1>
14 Type=linux_raw
15 Proto=Ethernet
16 Role=Internal
17 </interface>
18
19 <IPList WebServers>
20 5.0.0.1/32
21 </list>
22
23 <action log>
24 response=alert console
25 response=alert file(/var/log/hogwash/
    hogwash.alert)
26 response=dump packet(/var/log/hogwash/
    packet.log)
27 </action>
28
29 <action drop>
30 response=alert console
31 response=alert file(hogwash.alert)
32 response=dump packet(packet.log)
33 response=drop
34 </action>
35
36 <routing>
37 MacFilter(eth0, eth1)
38 </routing>
```

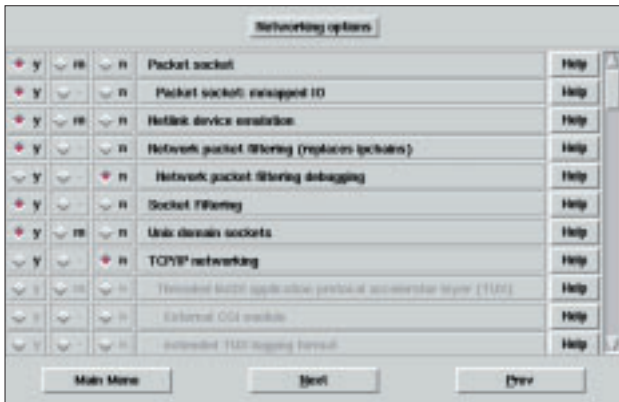


Abbildung 1: Beim Einsatz von Hogwash benötigt der Kernel keinen IP-Stack, man kann ihn komplett deaktivieren. Hogwash analysiert die Pakete selbst.

entfernen (siehe [Abbildung 1](#)): Damit ist der Host selbst hervorragend vor Angriffen geschützt. Allerdings kann selbst der Administrator nicht mehr mit TCP/IP auf den Hogwash-Rechner zugreifen, er muss das System lokal administrieren.

Konfiguration als Schrubber

Die Arbeitsweise von Hogwash wird von einem Konfigurationsfile und einer Regeldatei bestimmt. Das Paket enthält bereits die Beispiel-Konfigurationsfiles »stock.config« und »test.config« sowie mehrere Beispiel-Regeldateien:

```
general.rules
jason.rules
nikto.rules
nimda.rules
stock.rules
test.rules
x11.rules
```

Der Regelsatz »stock.rules« lädt zusätzlich noch die vier Dateien »nikto.rules«, »x11.rules«, »nimda.rules« und »general.rules«.

Für die Konfiguration muss der Administrator zunächst das lokale System, die Netzwerkkarten, IP-Adressen, Aktionen und Routing-Optionen definieren. [Listing 1](#) zeigt eine Beispieldatei für den Aufbau aus [Abbildung 2](#). Zunächst er-

hält der Hogwash-Sensor in dem Bereich »<system>« einen Namen und eine Identifikationsnummer. Die »Threads«-Angabe legt dann fest, ob Hogwash single-threaded (0) startet oder ob jede Netz Karte ihren eigenen Thread erhält. Zu beachten ist dabei, dass das System im Multi-threaded-Modus wesentlich performanter ist als in der anderen Variante.

Anschließend sind die Netzwerkkarten zu konfigurieren. Hogwash beherrscht mehrere Techniken, um mit den Interfaces zu kommunizieren: Linux (»Type = linux_raw«), OpenBSD (»osbsd_bpf«), MacOS X (»osx_bpf«) und Solaris (»solaris_dlpi«). Zusätzlich kann das System im Offline-Modus die Netzwerkpakete aus einer Tcpcdump-Datei lesen (»Type = tcpcdump«). Das einzige unterstützte Netzwerkprotokoll ist »Proto = Ethernet« – in Token-Ring-Netzwerken funktioniert Hogwash noch nicht. Die Angabe »Role = ...« ist optional, sie kommt für Routing und ARP zum Einsatz.

Um später beim Definieren der Regeln Arbeit zu sparen, empfiehlt es sich, in der Konfigurationsdatei IP-Adresslisten anzulegen. Sie enthalten einzelne IP-Adressen oder ganze Netzwerke in CIDR-Notation (Classless Internet Domain Routing). Zudem definiert der Admin die Aktionen, die Hogwash durchführen soll, wenn es Angriffe bemerkt:

- »response = alert console«, das Paket auf der Konsole protokollieren.
- »response = alert file(*Filename*)«, die Meldung in eine Datei schreiben.
- »response = dump packet(*Filename*)«, Paket zur späteren Analyse sichern.

Soll Hogwash das Paket auch verwerfen und so einen Rechner schützen, ist dafür als zusätzliche Reaktion »response = drop« einzustellen. Die Regeln greifen später auf die zu Aktionen zusammenfassten Reaktionen zu und führen bei einem Treffer die gewünschten Responses durch.

Arbeitet Hogwash als Gateway-IDS (häufig als Intrusion Prevention System bezeichnet), muss es die Pakete weiterleiten. Das bezeichnet Hogwash als Routing, obwohl es eigentlich als Bridge arbeitet. Neben dem einfachen Bridge-Modus »SBridge« kennt Hogwash eine Variante mit Flood Protection: »MacFilter«. Das System kann auch nur bestimmte IP-Adressen über einzelne Netzwerkkarten weiterleiten, abhängig von der Source-IP »SIP« oder der Destination-IP »DIP«. Die Bridge-Funktion erleichtert den Aufbau sehr, da Hogwash dann direkt nach dem Start ohne weitere IP-Konfiguration funktioniert. Das Gateway-IDS ist auf der IP-Ebene vollkommen transparent.

Streng nach Regel

Die Regeldatei in [Listing 2](#) stellt einige Beispielregeln vor. Jeder Eintrag kann sämtliche Aspekte des Pakets beachten: Netzwerkkarte, Ethernet-Typ (IP, ARP und so weiter), IP-Adressen, IP-Protokoll, ICMP-Code und -Typ, UDP- und TCP-Ports sowie Inhalte von TCP-Paketen. Als Reaktion auf ein Paket sind Nachrichten und Aktionen möglich: Beim Definieren einer Nachricht (»message = «) hat der Admin über Variablen Zugriff auf wichtige Eigenschaften des Pakets sowie auf das aktuelle Datum. Die Angabe der »action« bezieht sich auf die Aktionen, die in der Konfigurationsdatei ([Listing 1](#)) definiert sind.

[Listing 2](#) zeigt zwei sehr einfache Beispielregeln. Die erste prüft, ob ein TCP-Paket an Port 80 gerichtet ist und als In-

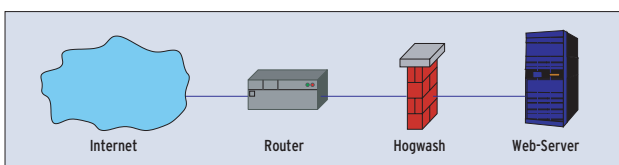


Abbildung 2: Im Normalfall arbeitet Hogwash wie eine Bridge: Weder Router noch Webserver bemerken in dieser Anordnung die zusätzliche Schranke. Erst wenn der Filter einen Angriff erkennt, blockiert er diese Verbindung.

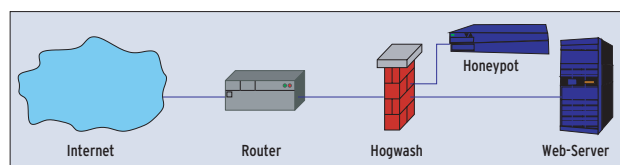


Abbildung 3: Hogwash schützt im Bait'n'Switch-Modus einen Produktionsserver und lenkt Angriffe auf den Honeypot um. So kann der Admin seine Widersacher in Ruhe beobachten, ohne Produktsysteme zu gefährden.

halt die Buchstabenfolge »chunked« enthält. Dies ist ein Indiz für einen Angriff auf die Apache Chunked-Encoding Memory Corruption Vulnerability [5]. Die zweite Regel prüft, ob es sich um einen ICMP-Echorequest handelt. In diesem Fall protokolliert Hogwash das Paket.

Die sehr komfortable Regelsprache erlaubt bereits die Definition beliebiger Regeln. Zusätzlich wird eine kommende Hogwash-Versionen einen Kompatibilitätsmodus für die Snort-Regelsprache enthalten. Dann wird es möglich sein, Snort-Regeln direkt zu übernehmen.

Bait'n'Switch

Der Bait'n'Switch-Modus erlaubt es, einen Produktionsserver mit einem Honeygot zu schützen. Der Hogwash-Rechner vermittelt dabei – wie **Abbildung 3** darstellt – zwischen dem Netzzugang, dem Honeygot und dem Produktionsser-

ver. Die passende Konfiguration ist in **Listing 3** auszugsweise zu sehen.

Der Admin definiert zunächst in den IP-Listen, welche Rechner Hogwash immer zum Server »<IPList Gruen>« und welche zum Honeygot »<IPList Rot>« leiten soll. Anschließend wählt er bei den Regeln die Aktion »Route«, um die Verbindung an den Honeygot umzuleiten. Die Aktion enthält einen Timeout sowie die IP-Adressen, die Hogwash nie auf den Honeygot leiten soll (»response = bns(Timeout,Adressliste«). Dann ist zu definieren, welche Netzwerkkarte wofür zuständig ist: »bns(Internet,Produktion, Honeygot,Rot-Liste)«.

Die Konfiguration des Honeygot sollte identisch sein mit dem Produktionsserver. Wichtig ist, dass die IP-Adressen übereinstimmen. Zusätzlich sollte der Honeygot möglichst identische Daten wie der echte Server enthalten, um Angreifer konsequent zu täuschen.

Listing 2: Regeldatei

```
01 <rule>
02 ip dst(WebServers)
03 tcp dst(80)
04 tcp nocase(chunked)
05 message=%sip-%dip Apache Chunked Encoding Attack
   Bugtraq 5033
06 action=drop
07 </rule>
08
09 <rule>
10 icmp type(8)
11 message=%sip-%dip icmp echo request
12 action=log
13 </rule>
```

Listing 3: Bait'n'Switch-Modus

```
01 <IPList Gruen>
02 x.x.x.x/24
03 </list>
04
05 <IPList Rot>
06 y.y.y.y/32
07 </list>
08
09 <Action Route>
10 response=alert console
11 response=alert file(/var/log/hogwash/hogwash.alert)
12 response=dump packet(/var/log/hogwash/packet.log)
13 response=bns(3600,Gruen)
14 </Action>
15
16 <routing>
17 bns(eth0,eth1,eth2,Rot)
18 </routing>
```

Fazit

Hogwash ist ein einfach zu installierendes, unsichtbares Gateway-IDS. Es arbeitet auf Layer 2 und ist damit für den IP-Verkehr vollkommen transparent, ein Angreifer kann es mit »traceroute« nicht erkennen. Das eingesetzte Betriebssystem benötigt noch nicht einmal einen IP-Stack (**Abbildung 1**). Hogwash eignet sich auch als zentraler Bestandteil eines Bait'n'Switch-Systems, das Angreifer vom Produktionsrechner auf einen Honeygot umleitet. So kann ihn der Admin

in Ruhe beobachten und den Angriff später analysieren.

Ein ähnliches, aber aufwendiger zu installierendes Gateway-IDS ist Snort-Inline [6]. Es verwendet das Bridging-Patch von Lennert Buytenhek und leitet alle Pakete mit Hilfe des Iptables-Targets »QUEUE« an Snort-Inline weiter. Diese Software testet dann die Pakete und lässt nur die ungefährlichen zu. Snort-Inline wird Thema in einer der folgenden Ausgaben sein. (fjl) ■

Infos

- [1] Hogwash: [<http://hogwash.sourceforge.net>]
- [2] Snort: [<http://www.snort.org>]
- [3] Ralf Spenneberg: „Intrusion Detection für Linux-Server“, Markt und Technik, 2002
- [4] Ralf Hildebrand, „Kain und Abel – Snort und Nmap, zwei Seiten derselben Münze“: Linux-Magazin 12/00, S. 102
- [5] Apache-Sicherheitslücke: [<http://www.securityfocus.com/bid/5033/exploit/>]
- [6] Snort-Inline [<http://www.snort.org/dl/contrib/patches/inline/>]
- [7] Securing an unpatchable Webserver... Hogwash!: [<http://www.securityfocus.com/infocus/1208>]

Der Autor

Ralf Spenneberg arbeitet als freier Unix/Linux-



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

Historie

Jed Haile und Jason Larson standen 1996 vor einem Problem: Einer ihrer Webserver hatte eine Sicherheitslücke, war aber eng mit einer Zusatzsoftware verzahnt. Nach einem Security-Update hätte Letztere nicht mehr funktioniert. Als Lösung schrieben sie den Hogwash-Vorgänger Scrub: Das Programm filterte exakt jene Pakete aus dem Netzwerkverkehr, die für den Einbruch auf dem Webserver genutzt wurden. So konnten die Admins ihren Webserver weiter betreiben, bis die Anwendung auf den neuen Webserver portiert war.

Drei Jahre später lernten sie Snort kennen. Sie waren fasziniert von dessen einfachem Aufbau und der leicht verständlichen Regelsprache. Daher integrierten sie die interne Snort-Maschine in Scrub und nannten ihr Programm fortan Snortscrub.

Schließlich gaben sie dem Produkt ein weiteres Mal einen neuen Namen: Hogwash. Sie wollten damit auch eine kommerzielle Verwendung ermöglichen. Hierauf folgte eine Phase von etwa zwei Jahren, in denen die Weiterentwicklung der frei verfügbaren Version ins Stocken kam. Das war um so mehr der Fall, je mehr Funktionen in Snort implementiert wurden. Die Snort-Detektionsmaschine zeigte ihre Schwächen.

So entschloss sich Jason Larson (auch als Anon Poet bekannt), die originale Scrub-Detektionsmaschine zu überarbeiten und als H2 wieder zum Leben zu erwecken. H2 weist einen Kompatibilitätsmodus auf, der Snort-Regeln versteht. Bislang ist die Kompatibilität aber noch nicht vollständig. Die aktuelle Version Devel-0.5 nutzt bereits H2.