

# Versuch 5 - Snort - Intrusion Detection System (IDS)

Jerome Badt, Joshua Hill, Dennis Tobias Rautenberg

November 16, 2016

# Chapter 1

## Intro

### 1.1 Begriffserklärung - Intrusion Detection System (IDS)

Als Intrusion-Detection wird die aktive Überwachung von Computersystemen und/oder -netzen mit dem Ziel der Erkennung von Angriffen und Missbrauch bezeichnet. Das Ziel von Intrusion-Detection besteht darin, aus allen im Überwachungsbereich stattfindenden Ereignissen diejenigen herauszufiltern, die auf Angriffe, Missbrauchsversuche oder Sicherheitsverletzungen hindeuten, um diese anschließend vertieft zu untersuchen. Ereignisse sollen dabei zeitnah erkannt und gemeldet werden.

Wir wollen zur Überwachung einen sog. netzbasierten Sensor aufsetzen: Netzbasierte Sensoren (Netzsensoren) überwachen den Netzverkehr eines Rechners oder eines ganzen Teilnetzes auf verdächtige Ereignisse. Zum Betrieb jedes Netzsensors wird typischerweise ein separater Rechner eingesetzt, so dass andere Applikationen nicht gestört werden können. Teilweise liefern Hersteller Netzsensoren nur noch in Kombination mit der zugehörigen Hardware/Software-Plattform, als so genanntes Appliance.

### 1.2 Aufgabenablauf

#### 1.2.1 1. Teil: Live-Hacking

Der Betreuer wird einen Angriff vorführen. Um welche Angriffe es sich handelt, wird erst am Übungstag entschieden.

#### 1.2.2 2. Teil: Konfiguration des IDS

##### Snort Konfiguration - Allgemein

Die Konfigurationsdateien und Angriffssignaturen befinden sich im Verzeichnis `/etc/snort/`. Die zentrale Konfigurationsdatei heißt `/etc/snort/snort.conf`, die Angriffssignaturen befinden sich in den Dateien mit der Endung `.rules` im Ordner `/etc/snort/rules/`, sie werden von der zentralen Konfigurationsdatei eingebunden. Die Snort Angriffssignaturen bestehen aus mehreren Teilen:

- Angabe von Quelle und Ziel: Hier werden Quell- und Zieladresse, sowie Quell- und Zielpport, sowie das Protokoll angegeben.
- Angabe von Paketeigenschaften und Inhalten: Hier werden z. B. Bytesequenzen aus den Paketen oder TCP Flags angegeben.
- Aktionsteil: Hier wird angegeben, was bei Erkennen der spezifizierten Angriffssignatur unternommen werden soll.

## Chapter 2

# Selbstversuch

### 2.1 Snort

#### 2.1.1 Installation

Die Installation von Snort über die Kommandozeile:

```
apt install snort
```

Das Interface und die IP Range muss vorher mit ifconfig abgefragt werden:

```
ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
inet6 fe80::d071:9000:a1b0:2e7d prefixlen 64 scopeid 0x20<link>
ether 08:00:27:03:78:5e txqueuelen 1000 (Ethernet)
RX packets 10 bytes 4899 (4.8 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 74 bytes 8194 (8.1 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1 (Local Loopback)
RX packets 2279 bytes 139877 (139.8 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 2279 bytes 139877 (139.8 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Während der Installation von snort, finden bezüglich dem Interface und der IP Range 2 Abfragen statt:

```
sudo apt install snort
Reading package lists... Done
Building dependency tree
Reading state information... Done
snort is already the newest version (2.9.7.0-5).
0 to upgrade, 0 to newly install, 0 to remove and 92 not to upgrade.
1 not fully installed or removed.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] Y
Setting up snort (2.9.7.0-5) ...
```

In unserem Beispiel haben wir das Netzwerk Interface enp0s3 eingetragen und das IP Subnetz (141.62.66.0/24) des Labors angegeben.

### 2.1.2 Konfiguration

Im Konfigurationsfile von Snort (das unter /etc/snort/snort.conf zu finden ist), haben wir lediglich eine zusätzliche Zeile eingetragen:

```
output alert_fast: alert.log
```

## 2.2 Teil 1 - Hacking

In unserem Versuch wurden automatisierte Pentests seitens des Dozenten durchgeführt, die um die 40000 Angriffe umfassten. Einen einzelnen Angriff heraus zu filtern, ist dadurch schwierig. Wir versuchen, einen konkreten Angriff genauer zu beleuchten.

### 2.2.1 TCP-SYN-Scan

Beim TCP-SYN-Scan wird ein TCP-Paket mit SYN-Flag an den Ziel-Host gesendet, um einen Verbindungsversuch vorzutäuschen. Die Antwort des Hosts gibt Aufschluss über den Port: Sendet er ein SYN/ACK-Paket, den zweiten Teil des Drei-Wege-Handshakes von TCP, akzeptiert der Port Verbindungen und ist daher offen. Der Quell-Host antwortet dann in der Regel mit einem RST-Paket, um die Verbindung wieder abzubauen (dies geschieht meist allerdings nicht durch den Portscanner, sondern durch das Betriebssystem, da offiziell kein Verbindungsversuch unternommen wurde). Sendet der Host ein RST-Paket, ist der Port geschlossen. Sendet der Ziel-Host überhaupt kein Paket, ist ein Paketfilter vorgeschaltet.<sup>1</sup>

Das passiert im alert.log, wenn man Port 0 scannt (sehr typisch bei einem NMAP Port Scan Port 0 mit anzugeben):

```
11/11-11:26:43.073895  [**] [1:524:8] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3] {TCP} 141.62.66.190:41557 ->
141.62.66.111:0
```

---

<sup>1</sup><https://de.wikipedia.org/wiki/Portscanner#TCP-SYN-Scan>

attack1.pcapng

Datei Bearbeiten Ansicht Navigation Aufzeichnen Analyse Statistiken Telephone Wireless Tools Hilfe

Anzeigefilter anwenden ...<Ctrl-F>

No.	Time	Source	Destination	Protocol	Length	Info
1548.	1218.881666	141.62.66.125	141.62.66.109	TCP	60	56735→59735 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1549.	1218.881671	141.62.66.109	141.62.66.184	TCP	60	59735→5002 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1549.	1218.882018	141.62.66.109	141.62.66.186	TCP	60	59735→8051 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1549.	1218.882269	141.62.66.184	141.62.66.109	TCP	60	5002→59735 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1549.	1218.882274	141.62.66.109	141.62.66.190	TCP	60	59735→9002 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1549.	1218.882551	141.62.66.109	141.62.66.207	TCP	60	59735→14571 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1549.	1218.882770	141.62.66.207	141.62.66.109	TCP	60	14571→59735 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1549.	1218.882772	141.62.66.109	141.62.66.208	TCP	60	59735→1688 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1549.	1218.883051	141.62.66.190	141.62.66.109	TCP	60	5002→59735 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1549.	1218.883056	141.62.66.200	141.62.66.109	TCP	60	1688→59735 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1549.	1218.883058	141.62.66.109	141.62.66.213	TCP	60	59735→20201 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1549.	1218.883330	141.62.66.109	141.62.66.214	TCP	60	59735→32771 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1549.	1218.883895	141.62.66.109	141.62.66.233	TCP	60	59735→99 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1549.	1218.884150	141.62.66.213	141.62.66.109	TCP	60	20201→59735 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1549.	1218.884162	141.62.66.109	141.62.66.235	TCP	60	59735→5432 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1549.	1218.884163	141.62.66.233	141.62.66.109	TCP	60	99→59735 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1549.	1218.884395	141.62.66.235	141.62.66.109	TCP	60	5432→59735 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1549.	1218.884397	141.62.66.214	141.62.66.109	TCP	60	32771→59735 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

> Frame 154093: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0  
 > Ethernet II, Src: Raspberr\_49:75:6c (b8:27:eb:49:75:6c), Dst: HewlettP\_Sf:31:ba (e8:39:35:5f:31:ba)  
 > Internet Protocol Version 4, Src: 141.62.66.125, Dst: 141.62.66.109  
 > Transmission Control Protocol, Src Port: 56737, Dst Port: 59735, Seq: 1, Ack: 1, Len: 0  
 Source Port: 56737  
 Destination Port: 59735  
 [Stream index: 60202]  
 [TCP Segment Len: 0]  
 0000 e8 39 35 5f 31 ba 00 27 e8 39 35 6c 00 00 40 00 .95.1...Jul..t.  
 0010 00 28 c9 c7 40 00 40 06 d1 b1 ba 3e 42 7d 8d 3e (-.-.-.-.->  
 0020 42 6d dd a1 e9 57 00 00 00 00 11 e0 ea c7 50 14 8m...M.....P.  
 0030 00 00 4c c8 00 00 00 00 00 00 00 00 ..L.....

Figure 2.1: TCP Syn Scan

11/11-11:26:43.074293    [\*\*] [1:524:8] BAD-TRAFFIC tcp port 0 traffic [\*\*]  
 [Classification: Misc activity] [Priority: 3] {TCP} 141.62.66.111:0 ->  
 141.62.66.190:41557

## 2.3 Teil 2 - IDS

### 2.3.1 Snort Regeln

#### Aufgabe

- Schreiben Sie eine Regel, die auf TCP SYN Pakete auf Port 111 anspricht!
- Versuchen Sie eine Regel zu schreiben, die anspricht, wenn ein Paket die Bytesequenz `istestlabor` enthält! (Zum testen können Sie ja eine entsprechende `index.html` Datei für den Webserver erstellen oder diese Sequenz als ftp login probieren.)

#### Vorgehen

Öffnen der rules-Datei:

```
sudo gedit /etc/snort/rules
```

Eintragen der neuen Regeln:

```
alert tcp any any -> any 111
(flags: S; msg: "Possible SYN FIN scan"; sid:10000001;)
alert tcp any any -> any any
(msg: "String istestlabor detected"; content:"istestlabor" sid:10000002;)
```

- `tcp/udp`: Gibt an, welcher Pakettyp verwendet wird.
- `Externe IP & Externer Port -> Interne IP Interner Port`
- `flags: S` steht für SYN Pakete. (Siehe Three-Way-Handshake).  
Es können auch andere Pakete selektiert werden, auf die reagiert werden soll.
- `msg`: Gibt an, welche Nachricht im `alert.log` erscheinen soll, falls ein Muster detektiert wird.
- `content`: Gibt an, auf welchen String reagiert werden soll.
- `sid`: ist ein Akronym für Snort ID. Jeder Snort Regelsatz hat eine einzigartige ID die es Output Modulen oder Log Scannern ermöglicht, die Regel die den Alarm ausgelöst hat zu identifizieren.

Getestet wurde, in dem wir eine FTP Verbindung zu einem anderen Client aufgenommen haben und als User und Passwort `istestlabor` eingetragen haben. Die andere Rule wurde durch den Port Scan des Dozenten ausgelöst. Dannach haben wir kontrolliert, ob die Regel auch tatsächlich funktioniert:

```
sudo gedit /var/log/snort/alert.log
```

```
11/11-11:54:44.087925  [**] [1:1000002:0] String istestlabor detected [**]
[Priority: 0] {TCP} 141.62.66.104:53724 -> 141.62.66.103:21
11/11-12:06:35.235065  [**] [1:1000001:0] Possible SYN FIN Scan [**]
[Priority: 0] {TCP} 141.62.66.190:56276 -> 141.62.66.102:111
```