Network and Security Management

NIDS

Projekt

Autor: Richard Grünert

5.6.2023

Inhaltsverzeichnis

1 Einführung

Network Intrusion Detection Systems (NIDS) sind Anwendungen, die versuchen, schädliche Aktivitäten im Netzwerk zu erkennen, indem sie dessen gesamten Traffic aufzeichnen und analysieren. Ein beliebtes NIDS-Programm ist das free/open-source tool **snort**, welches hier zum Einsatz kommt. Es bietet ein modulares Konfigurationssystem über sog. *rules* für Traffic- und Protokollanalysen sowie die Erkennung verschiedenster Angriffe über Signaturmatching.

Dieses Dokument beschreibt die Installation einer Testumgebung für snort auf Basis von VirtualBox, welche anschließend durch verschiedene Angriffe getestet wird. Die Netzwerkstruktur zeigt die folgende Abb. 1.

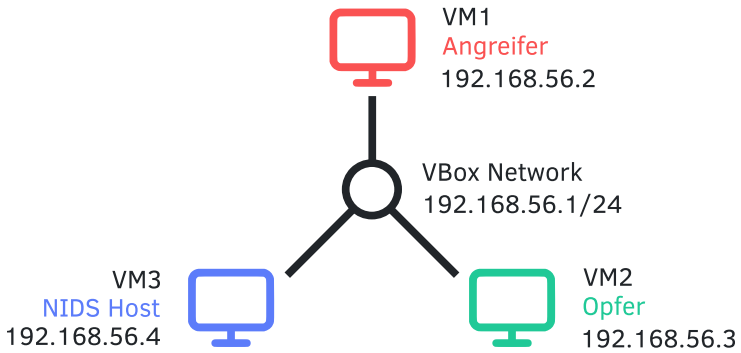

Abbildung 1: Virtuelle Netzwerkstruktur.

Die drei virtuellen Maschinen befinden sich im gleichen virtuellen Netzwerk. Ein Angreifersystem (Kali Linux) versucht, Hosts im Netzwerk mit unterschiedlichen Methoden anzugreifen. Der NIDS-Host sollte dann mit snort diese Aktivitäten erkennen.

Zunächst folgt das Setup der Testumgebung.

2 Setup

2.1 Installation der Virtuellen Maschinen

Die drei virtuellen Maschinen werden unter Oracle VirtualBox entsprechend Tabelle 1 installiert. Die Installationen erfolgen mit den jeweiligen grafischen Installern, wobei grundsätzlich die Standardeinstellungen verwendet werden. Für die beiden Debian-Systeme wure keine Desktopoberfläche ausgewählt (lediglich `standard system utilities`). Abb. 2 zeigt die virtuellen Maschinen unter Virtualbox.

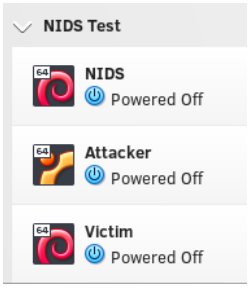

Abbildung 2: Verwendete Maschinen in Virtualbox.

2.2 Snort-Installation

Als nächstes wird snort auf der NIDS-VM installiert.

```
su
apt install snort
```

Bei der Installation erscheint ein Konfigurationsfenster. In dieses muss die CIDR-Adresse des VirtualBox-Netzwerkes (`192.168.56.0/24`) eingetragen werden (Abb. 3).

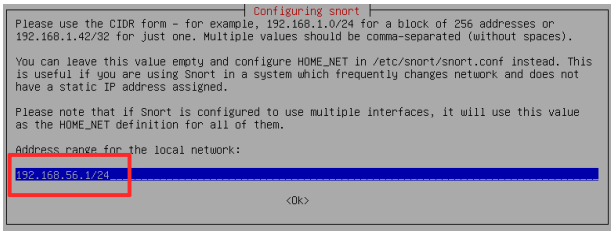

Abbildung 3: Snort Konfiguration bei der Installation.

Mit dem folgenden Befehl kann die snort-Installation überprüft werden.

```
/sbin/snort --version
```

2.3 Webserver-Installation

`nginx` wird auf dem Victim-Host als Ziel für den Angreifer installiert. Dazu wird lediglich folgender Befehl ausgeführt.

```
su
apt install nginx
```

2.4 Erstellen des Virtuellen Netzwerkes

Um das virtuelle Netzwerk einzurichten, öffnet man zunächst `File -> Tools -> Network Manager` und erstellt ein neues Netzwerk mit der Schaltfläche `Create` (Abb. 4).

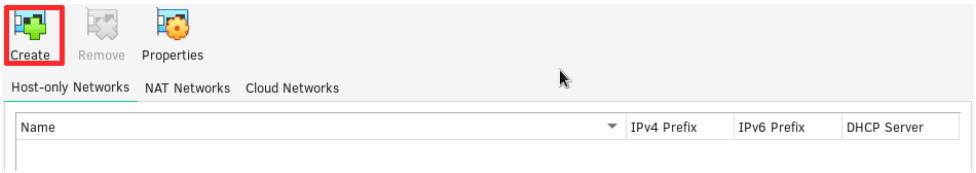

Abbildung 4: Erstellen des virtuellen Netzwerkadapters

Danach kann man über `Properties` die in Abb. 5 zu sehenden Einstellungen eintragen.

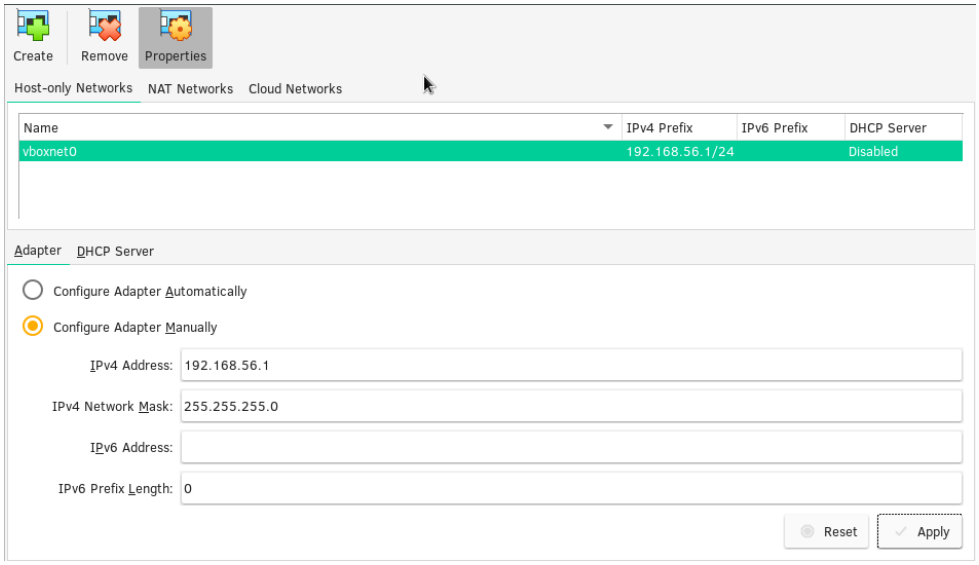

Abbildung 5: Einstellungen des Netzwerkadapters

Um die drei Hosts mit dem neuen virtuellen Adapter zu verbinden, wählt man die jeweilige Maschine aus, öffnet `Settings` und wechselt zum `Network`-Tab. Danach wählt man `Adapter 2`, aktiviert ihn, setzt `Attached to:` auf `Host-only Adapter` und wählt den vorher erstellten Adapter aus (Abb. 6, 7)

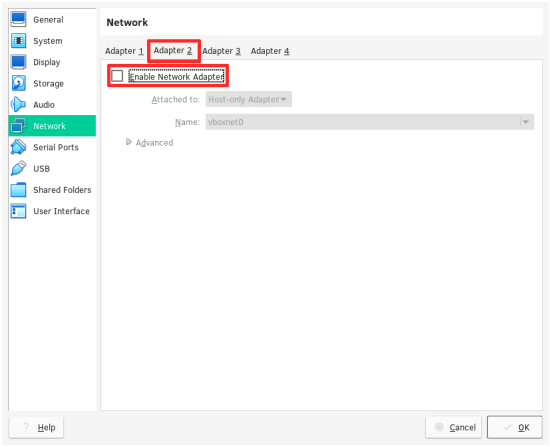

Abbildung 6: Aktivieren von Adapter 2.

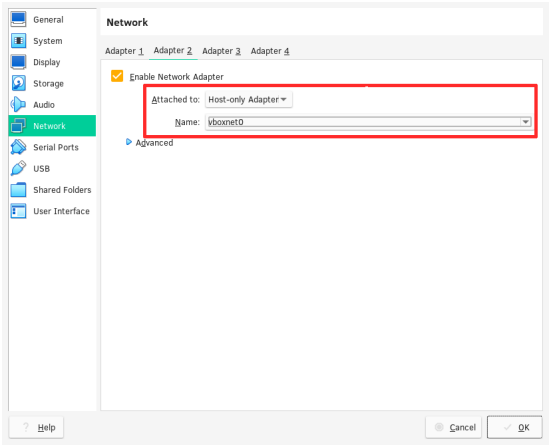

Abbildung 7: Auswählen des Adapters.

Damit snort den Netzwerktraffic anderer Hosts sniffen kann, muss zuletzt für die NIDS-Maschine der Promiscuous-Modus aktiviert werden (Abb. 8).

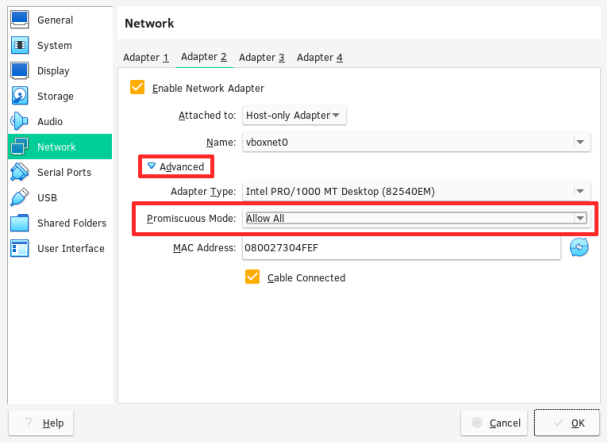

Abbildung 8: Promiscuous Mode für den NIDS-Host.

2.5 IP-Adressen-Vergabe

Jeder Host erhält eine statische IP-Adresse aus Reproduzierbarkeitsgründen. Diese können Tabelle 1 entnommen werden. Hierzu muss zuerst die Adapterbezeichnung auf der jeweiligen Maschine herausgefunden werden. Dies gelingt mit dem Befehl `ip addr show` (Abb. 9)

```
nids@NIDS:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1
000
    link/ether 08:00:27:5f:4a:65 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
       valid_lft 86400sec preferred_lft 86400sec
    inet6 fe80::a00:27ff:fe5f:4a65/64 scope link
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 08:00:27:30:4f:ef brd ff:ff:ff:ff:ff:ff
```

Abbildung 9: VM-Netzwerkinterfaces.

Danach muss die Datei `/etc/network/interfaces` entsprechend Abb. 10 bearbeitet werden und der `networking` Service neugestartet werden.

```
source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet dhcp

auto enp0s8
iface enp0s8 inet static
address 192.168.56.4
netmask 255.255.255.0
gateway 192.168.56.1
~
~
```

Abbildung 10: `/etc/network/interfaces`.

ID	OS	Hostname	user	password	IP
VM1	Kali Linux	Attacker	attacker	0000	192.168.56.2
VM2	Debian	Victim	victim	0000	192.168.56.3
VM3	Debian	NIDS	nids	0000	192.168.56.4

Tabelle 1: Zusammenfassung der Host-Konfigurationen

```
$ sudo systemctl restart networking.service
```

Wurde dies für alle VMs durchgeführt, kann ein einfacher Verbindungstest über den `ping`-Befehl ausgeführt werden.

3 NIDS-Konfiguration und Angriffe

3.1 Snort-Konfiguration

Die Hauptkonfigurationsdatei für snort liegt unter `/etc/snort/snort.conf`. In diesem Fall muss allerdings zuerst die Debian-spezifische Datei `/etc/snort/snort.debian.conf` entsprechend Abb. 11 bearbeitet werden. Das Feld `DEBIAN_SNORT_INTERFACE` wird entsprechend Abschnitt 2.5 gesetzt.

```
# snort.debian.config (Debian Snort configuration file)
#
# This file was generated by the post-installation script of the snort
# package using values from the debconf database.
#
# It is used for options that are changed by Debian to leave
# the original configuration files untouched.
#
# This file is automatically updated on upgrades of the snort package
# *only* if it has not been modified since the last upgrade of that package.
#
# If you have edited this file but would like it to be automatically updated
# again, run the following command as root:
#   dpkg-reconfigure snort

DEBIAN_SNORT_STARTUP="boot"
DEBIAN_SNORT_HOME_NET="192.168.56.0/24"
DEBIAN_SNORT_OPTIONS=
DEBIAN_SNORT_INTERFACE="enp0s8"
DEBIAN_SNORT_SEND_STATS="true"
DEBIAN_SNORT_STATS_RCPT="root"
DEBIAN_SNORT_STATS_THRESHOLD="1"
```

Abbildung 11: `/etc/snort/snort.debian.conf`

Es gibt einige vorgefertigte Regeln, welche unter `/etc/snort/rules` zu finden sind. In `/etc/snort/snort.conf` findet man am Ende der Datei alle Regeln (Pfade), die snort einbeziehen soll, d.h die Regeln, die tatsächlich aktiv sind.

Um die Konfiguration nach dem Bearbeiten auf Fehler zu prüfen, kann man folgenden Befehl nutzen.

```
/sbin/snort -T -i enp0s8 -c /etc/snort/snort.conf
```

Info

Snort kann über die Tastenkombinaition `Ctrl+z` beendet werden.

3.2 Benutzerdefinierte Regeln

Die Regeldefinitionen folgen einer einfachen Syntax. Die grundlegende Struktur zeigt Abb. 12. Als Referenz eignen sich auch die vorgefertigten Regeln unter `/etc/snort/rules`.

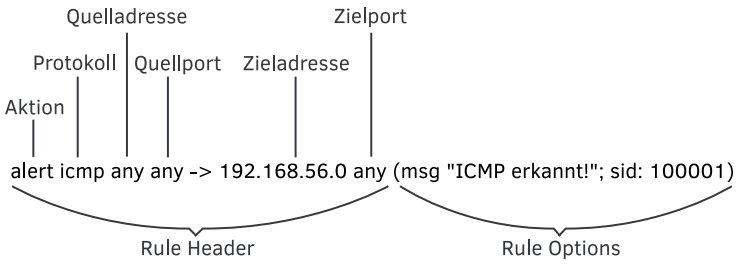

Abbildung 12: Struktur der Snort-Regeln.

Die dargestellte Regel alarmiert bei Erkennung eines ICMP-Paketes (z.B. ping). Um sie hinzuzufügen fügt man die Zeile aus Abb 12 in die zuerst leere Datei `/etc/snort/rules/local.rules` ein. Die Zieladresse kann auch durch die Umgebungsvariable `$HOME_NET` ersetzt werden (Abb. 13). [1]

```
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.

alert icmp any any -> $HOME_NET any (msg: "ICMP erkannt!"; sid: 100001)
```

Abbildung 13: `/etc/snort/rules/local.rules`

Dann startet man snort mit dem Befehl

```
1 /sbin/snort -q -l /var/log/snort -i enp0s8 -A console -c /etc/snort/snort.conf
```

Danach kann vom Angreifer ein `ping`-Befehl auf den Victim-Host durchgeführt werden, welcher wie in Abb. 14 durch snort erkannt werden sollte.

```
06/04-14:02:38.763830  [**] [1:100001:0] ICMP erkannt! [**] [Priority: 0] {ICMP} 192.168.56.2 -> 192
.168.56.3
```

Abbildung 14: Ergebnis des Ping-Tests.

Als Hilfe zur Erstellung von Regeln eignet sich die Website *snorpy* [1], welche eine grafische Oberfläche bietet, um Regeln zu erstellen und mögliche Optionen zu erkunden.

3.3 nmap

Mit der Standardkonfiguration lässt sich ein einfacher `nmap`-Scan sofort erkennen.
Vom Angreifer aus:

```
1 nmap 192.168.56.3
```

[1] Die `sid` kann quasi ein beliebiger einzigartiger Wert sein.

```
┌──(attacker㉿Attacker)-[~]
└─$ nmap 192.168.56.3
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-04 15:10 CEST
Nmap scan report for 192.168.56.3
Host is up (0.00045s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 13.10 seconds
```

Abbildung 15: Ergebnis von nmap beim Angreifer.

Das Resultat des NIDS zeigt Abb. 16.

```
06/04-14:48:58.673636  [**] [1:1418:11] SNMP request tcp [**] [Classification: Attempted Information
 Leak] [Priority: 2] {TCP} 192.168.56.2:55134 -> 192.168.56.3:161
```

Abbildung 16: NIDS altert bei nmap.

3.4 DoS-Angriff / SYN-Flood

Die Standardkonfiguration von snort kann bereits DoS-Angriffe erkennen. Für eine einfache benutzerdefinierte Konfiguration kann man die folgende Regel verwenden.

```
1 alert tcp any -> $HOME_NET 80 (flag: S; msg: "Potentieller DoS-Angriff!"; sid: 100002)
```

Hierbei wird die Option `flag: S` genutzt, um TCP-Pakete zu erkennen, bei denen nur das SYN-Flag gesetzt ist. Weitere Flags finden sich in der snort-Dokumentation [2].

Der Webserver auf dem Victim-Host sollte bereits mit einer Beispielseite laufen. Dies kann z.B. über Firefox auf dem Angreifer überprüft werden. Da der Webbrowser aufgrund von Caching ungeeignet ist für die Erkennung des Serverzustandes, kann man alternativ den folgenden Befehl nutzen.

```
1 curl -I 192.168.56.3
```

Dieser sollte bei erfolgreicher Bearbeitung durch den Server die in Abb. 17 zu sehende Ausgabe liefern.

```
└─$ curl -I 192.168.56.3
HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Mon, 05 Jun 2023 12:47:19 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Sun, 04 Jun 2023 13:03:53 GMT
Connection: keep-alive
ETag: "647c8bb9-264"
Accept-Ranges: bytes
```

Abbildung 17: Erfolgreiche Webserverantwort.

Das auf Kali vorinstallierte Tool `hping3` kann nun für einen einfachen TCP SYN-Angriff verwendet werden. Dazu wird folgender Befehl genutzt.

```
1 sudo hping3 --flood -S --rand-source -p 80 192.168.56.3
```

Nun kann die Seite erneut angefragt werden (z.B. auf dem Angreifer-System). Es sollte zu Verzögerungen kommen. Gleichzeitig sollte das snort-System die Anfragen als „Potentially Bad Traffic“ klassifizieren (Abb. 18). Als Variation kann man das Argument `--rand-source` im `hping3` weglassen. Sind keine Verzögerungen merkbar, kann der folgende alternative Befehl zu Anfrage möglicherweise helfen.

```
curl -I 192.168.56.3 -H "Cache-Control: no cache"
```

```
06/05-19:14:42.520615  [**] [1:503:7] MISC Source Port 20 to <1024 [**] [Clas
sification: Potentially Bad Traffic] [Priority: 2] {TCP} 166.172.253.167:20 -
> 192.168.56.3:80
```

Abbildung 18: SYN-Angriffserkennung durch snort.

4 Wireshark

Die mit dem snort-Flag `l` generierten Logdateien enthalten nicht die Alarmierungsnachrichten der Konsole, sondern sind Logs des gesamten Traffics. Praktischerweise können diese mit Wireshark geöffnet und nachträglich analysiert werden. Mithilfe der Konsolenausgabe oder eines externen Logs (über Option `-A fast`) können dann z.B. die Zeiten der Alarme genutzt werden, um die Angriffe in Wireshark ausfindig zu machen. Abb. 19 zeigt einen Ausschnitt der SYN-Flood aus Abschnitt 3.4.

No.	Time	Source	Destination	Proto	Length	Info
1713	2.260041	127.143.104.179	192.168.56.3	TCP	60	15775 → 80 [SYN] Seq=0 Win=512 Len=0
1714	2.260104	127.219.69.186	192.168.56.3	TCP	60	15778 → 80 [SYN] Seq=0 Win=512 Len=0
1715	2.261320	127.124.91.162	192.168.56.3	TCP	60	15849 → 80 [SYN] Seq=0 Win=512 Len=0
1716	2.261758	127.138.27.107	192.168.56.3	TCP	60	15877 → 80 [SYN] Seq=0 Win=512 Len=0
1717	2.262542	127.70.61.202	192.168.56.3	TCP	60	15927 → 80 [SYN] Seq=0 Win=512 Len=0
1718	2.262573	127.110.91.184	192.168.56.3	TCP	60	15928 → 80 [SYN] Seq=0 Win=512 Len=0
1719	2.262602	127.4.192.151	192.168.56.3	TCP	60	15930 → 80 [SYN] Seq=0 Win=512 Len=0
1720	2.263165	127.223.209.113	192.168.56.3	TCP	60	15963 → 80 [SYN] Seq=0 Win=512 Len=0
1721	2.264943	127.192.168.38	192.168.56.3	TCP	60	16075 → 80 [SYN] Seq=0 Win=512 Len=0
1722	2.266408	127.77.103.104	192.168.56.3	TCP	60	16168 → 80 [SYN] Seq=0 Win=512 Len=0
1723	2.269464	127.43.203.71	192.168.56.3	TCP	60	16412 → 80 [SYN] Seq=0 Win=512 Len=0
1724	2.271177	127.19.110.115	192.168.56.3	TCP	60	16551 → 80 [SYN] Seq=0 Win=512 Len=0
1725	2.272268	127.71.32.198	192.168.56.3	TCP	60	16645 → 80 [SYN] Seq=0 Win=512 Len=0
1726	2.274258	127.80.87.127	192.168.56.3	TCP	60	16813 → 80 [SYN] Seq=0 Win=512 Len=0
1727	2.276752	127.46.49.170	192.168.56.3	TCP	60	17018 → 80 [SYN] Seq=0 Win=512 Len=0
1728	2.276860	127.124.187.255	192.168.56.3	TCP	60	17029 → 80 [SYN] Seq=0 Win=512 Len=0
1729	2.277567	127.87.227.252	192.168.56.3	TCP	60	17088 → 80 [SYN] Seq=0 Win=512 Len=0
1730	2.280719	127.1.81.70	192.168.56.3	TCP	60	17260 → 80 [SYN] Seq=0 Win=512 Len=0
1731	2.280958	127.115.236.184	192.168.56.3	TCP	60	17281 → 80 [SYN] Seq=0 Win=512 Len=0
1732	2.281110	127.197.85.193	192.168.56.3	TCP	60	17294 → 80 [SYN] Seq=0 Win=512 Len=0
1733	2.281485	127.203.162.252	192.168.56.3	TCP	60	17324 → 80 [SYN] Seq=0 Win=512 Len=0
1734	2.281626	127.11.172.186	192.168.56.3	TCP	60	17336 → 80 [SYN] Seq=0 Win=512 Len=0
1735	2.282114	127.177.154.103	192.168.56.3	TCP	60	17373 → 80 [SYN] Seq=0 Win=512 Len=0
1736	2.285011	127.143.69.202	192.168.56.3	TCP	60	17551 → 80 [SYN] Seq=0 Win=512 Len=0
1737	2.285304	127.164.64.144	192.168.56.3	TCP	60	17569 → 80 [SYN] Seq=0 Win=512 Len=0

Abbildung 19: Snort Log in Wireshark

5 Zusammenfassung

Die dargestellten Beispiele bieten einen einfachen Einstieg in snort und die Funktionsweise von dessen Regeln. Komplexere Beispiele mit gezielten Attacken auf Vulnerabilitäten (z.B. MS EternalBlue) und deren Erkennung durch das NIDS könnten in einer fortführenden Arbeit behandelt werden.

Referenzen

[1] Christopher Davis. *Snorpy: A Web Based Snort Rule Creator / Maker for Building Simple Snort Rules*. URL: http://snorpy.cyb3rs3c.net/ (besucht am 22. 05. 2023).

[2] Snort Documentation. *Snort 3 Rule Writing Guide: flags*. URL: https://docs.snort.org/rules/options/non_payload/flags (besucht am 01. 06. 2023).