Aufbau & Analyse von Industrie-Netzwerken

Lerneinheit 2: Arbeiten mit Scapy

Prof. Dr. Christoph Karg

Studiengang Informatik
Hochschule Aalen

Sommersemester 2016

Gliederung

Diese Lernheinheit beschäftigt sich mit Scapy, einem Python-Werkzeug zur Analyse und Manipulation von Netzwerk-Kommunikation.

Sie gliedert sich in folgende Abschnitte:

1. Installation von Scapy
2. Arbeit mit Scapy in der Konsole
3. Erstellen von Scapy Skripten
4. DNS Spoofing
5. ARP Cache Poisoning

Scapy

- Scapy ist ein in Python geschriebenes Werkzeug zur Netzwerkanalyse.
- Scapy setzt Python 2.7.x voraus.
- Es wird an einem Port für Python 3 gearbeitet (Scapy3k)
- Scapy kann über die Kommandozeile bedient werden.
- Scapy stellt eine API für die Entwicklung eigener Python Skripte bereit.

Installation von Scapy

1. Installatieren von Python 2.7 und VirtualEnv:

```
> apt-get install python2.7 python-dev
```

2. Einrichten einer virtuellen Python Umgebung (optional):

```
> apt-get install virtualenv
> virtualenv scapy-ve
> source scapy-ve/bin/activate
```

3. Installieren von Scapy:

```
> pip install scapy pycrypto netifaces
```

Aufruf von Scapy

Befehl zum Starten von Scapy:

```
> sudo $HOME/scapy-ve/bin/scapy
```

Die Ausgabe sieht in etwa so aus:

```
INFO: Can't import python gnuplot wrapper. Won't be able to plot.
INFO: Can't import PyX. Won't be able to use psdump() or
pdfdump().
WARNING: No route found for IPv6 destination :: (no default
route?)
Welcome to Scapy (2.3.2)
>>>
```

Sniffen von Paketen

- Der Befehl zum Sniffen lautet `sniff()`.
- Nützliche Parameter:
  - `iface` $\rightsquigarrow$ zu nutzendes Netzwerk-Interface
  - `count` $\rightsquigarrow$ Anzahl der zu sniffenden Pakete
  - `filter` $\rightsquigarrow$ Filter zur Auswahl der Pakete
  - `prn` $\rightsquigarrow$ Funktion zur Auswertung eines Pakets
- Der Rückgabewert ist eine Statistik über die Art der gesnifften Pakete.

Sniffen von Paketen: Beispiele

Befehl:

```
>>> sniff(count=10)
```

Ausgabe:

```
<Sniffed: TCP:3 UDP:0 ICMP:0 Other:7>
```

Sniffen von Paketen: Beispiele (Forts.)

Definition einer Ausgabefunktion::

```
>>> def f(pkt):\
...     return pkt.summary()
...
```

Befehl:

```
>>> sniff(prn=f, count=5)
```

Ausgabe:

```
Ether / IP / TCP 162.125.17.3:https > 192.168.0.63:49952 PA / Raw
Ether / IP / TCP 192.168.0.63:49952 > 162.125.17.3:https PA / Raw
Ether / IP / TCP 162.125.17.3:https > 192.168.0.63:49952 A
Ether / ARP who has 192.168.0.69 says 192.168.0.1 / Padding
Ether / ARP who has 192.168.0.69 says 192.168.0.1 / Padding
<Sniffed: TCP:3 UDP:0 ICMP:0 Other:2>
```

Sniffen von Paketen: Beispiele (Forts.)

Befehl:

```
sniff(prn=f, count=10, filter="tcp")
```

Ausgabe:

```
Ether / IP / TCP 162.125.17.3:https > 192.168.0.63:49952 PA / Raw
Ether / IP / TCP 192.168.0.63:49952 > 162.125.17.3:https PA / Raw
Ether / IP / TCP 162.125.17.3:https > 192.168.0.63:49952 A
Ether / IP / TCP 192.168.0.63:38584 > 172.217.21.142:https A
Ether / IP / TCP 172.217.21.142:https > 192.168.0.63:38584 A
Ether / IP / TCP 198.252.206.25:https > 192.168.0.63:37662 PA / Raw
Ether / IP / TCP 192.168.0.63:37662 > 198.252.206.25:https PA / Raw
Ether / IP / TCP 198.252.206.25:https > 192.168.0.63:37662 A
Ether / IP / TCP 172.217.21.142:https > 192.168.0.63:38584 PA / Raw
Ether / IP / TCP 192.168.0.63:38584 > 172.217.21.142:https A
<Sniffed: TCP:10 UDP:0 ICMP:0 Other:0>
```

Ausgabe von Informationen

- Der Befehl `lsc()` listet alle zur Verfügung stehenden Befehle auf.
- Der Befehl `ls()` listet alle unterstützten Paketformate auf.
- Mit `ls(pkt)` wird der Inhalt des Pakets `pkt` ausgegeben.
- Ein Datenpaket besitzt folgende Methoden zur Ausgabe:
  - `summary()` $\rightsquigarrow$ Kurzinfo zum Paket ausgeben
  - `show()` $\rightsquigarrow$ kompletten Inhalt des Pakets ausgeben

Senden und Empfangen von Paketen

- Layer 3 (Internet Protokoll):
  - `send()` $\rightsquigarrow$ Senden von Paketen
  - `sr()` $\rightsquigarrow$ Senden von Paketen und Rückgabe der empfangenen und nicht beantworteten Pakete
  - `sr1()` $\rightsquigarrow$ Senden eines Pakets und Rückgabe des ersten Pakets der Antwort
- Layer 2 (Ethernet):
  - `sendp()` $\rightsquigarrow$ Senden von Paketen
  - `srp()` $\rightsquigarrow$ Senden und empfangen von Paketen

Konstruktion eines Datenpakets

Paketkonstruktion:

- Ein Datenpaket wird anhand seiner Komponenten zusammengebaut.
- Für jedes Paketformat existieren entsprechende Python Klassen.
- Das Gesamtpaket erhält man dir Konkatenation der Teilpakete.

Beispiel:

- Erstellen eines ICMP-Pakets:

```
pkt = IP(dst="192.168.0.72") / ICMP() / "Hello␣World"
```

- Erstellen eines weiteren Pakets mit korrekten Prüfsummen:

```
pkt2 = pkt.__class__(str(pkt))
```

Ausgabe des Datenpakets

```
###[ IP ]###
  version= 4L
  ihl= 5L
  tos= 0x0
  len= 39
  id= 1
  flags=
  frag= 0L
  ttl= 64
  proto= icmp
  chksum= 0xf8fd
  src= 192.168.0.63
  dst= 192.168.0.72
  \options\
###[ ICMP ]###
     type= echo-request
     code= 0
     chksum= 0xa631
     id= 0x0
     seq= 0x0
###[ Raw ]###
        load= 'Hello World'
```

Beispiel: Pingen eines Rechners

- Erstellen eines ICMP-Pakets:

```
pkt = IP(dst="192.168.0.72") / ICMP() / "Hello World"
```

- Versand des Pakets:

```
send(pkt)
```

- Versand des Pakets und Empfang der Antwort:

```
(ans,unans)=sr(pkt,retry=2,timeout=1)
ans.summary()
unans.summary()
```

- Versand des Pakets und Empfang des 1. Pakets der Antwort:

```
ans=sr1(pkt,retry=2,timeout=1)
ans.summary()
```

Beispiel: DNS-Anfrage

- Das Domain Name System dient zur Auflösung von Domainnamen zu IP-Adressen [Moc87].
- Für Anfragen wird das UDP-Protokoll verwendet.
- Mit Scapy lassen sich DNS-Anfragen einfach umsetzen, indem man ein passendes Paket konstruiert und versendet.
- Im folgenden Beispiel wird eine DNS-Anfrage an den Google DNS-Server mit der IP 8.8.8.8 gestellt.

Beispiel: DNS-Anfrage (Forts.)

Anfrage, um `www.google.de` aufzulösen:

```
query = DNS(rd=1,qd=DNSQR(qname="www.google.de"))
resp = sr1(IP(dst="8.8.8.8")/UDP()/query)
resp.show()
```

Anfrage, um die IPv6-Adresse `www.google.de` aufzulösen:

```
query = DNS(rd=1,qd=DNSQR(qname="www.google.de",qtype="AAAA"))
resp = sr1(IP(dst="8.8.8.8")/UDP()/query)
resp.show()
```

Traceroute mit Scapy

Befehl:

```
traceroute(["www.google.de"], maxttl=10)
```

Ergebnis:

```
Received 10 packets, got 10 answers, remaining 0 packets
   216.58.198.131:tcp80
1  192.168.0.1      11
2  10.217.248.1     11
3  172.30.2.69      11
4  84.116.190.49    11
5  84.116.190.2     11
6  84.116.134.9     11
7  84.116.140.189   11
8  213.46.177.42    11
9  216.239.59.60    11
10 216.239.59.60    11
(<Traceroute: TCP:0 UDP:0 ICMP:10 Other:0>,
<Unanswered: TCP:0 UDP:0 ICMP:0 Other:0>)
```

Weitere Funktionen

Neben den genannten Funktionen bietet Scapy diverse weitere:

- Lesen und Schreiben von pcap-Dateien
- Ausgabe von Datenpaketen als PDF-Dokument
- Erstellen von Plots mit GNU Plot
- Hex-Dumps von Paketen

Für weitere Details siehe [Bio10; Max12].

Erstellen von Scapy-Skripten

- Neben der Arbeit in der Konsole stellt Scapy auch eine API zur Programmierung von Python-Skripten zur Verfügung.
- Im folgenden werden vier Skripte erstellt:
  - Ein Sniffer für IP-Pakete
  - Ein Skript für DNS-Spoofing
  - Ein Sniffer für ARP-Pakete
  - Ein Skript für ARP Cache Poisoning

Live Hacking

Praktischer Teil

Literatur I

[Bio10] Philippe Biondi. *Scapy Documentation*. 19. Apr. 2010. URL: http://www.secdev.org/projects/scapy/doc/ (besucht am 30. 05. 2016).

[Max12] Adam Maxwell. *The Very Unofficial Dummies Guide To Scapy*. 2012. URL: https://github.com/catalyst256/scapy-guide/blob/master/ScapyGuide.pdf.

[Moc87] P. Mockapetris. *Domain Names - Implementation And Specification*. Internet Engineering Task Force. 1987. URL: https://www.ietf.org/rfc/rfc1035.txt.