

James Forshaw

# Netzwerk- protokolle hacken

Sicherheitslücken verstehen,  
analysieren und schützen



dpunkt.verlag



# Inhalt

**Cover**

**Titel**

**Impressum**

**Inhaltsübersicht**

**Vorwort**

**Danksagungen**

**Einführung**

**Inhaltsverzeichnis**

## **1 Netzwerk-Grundlagen**

1.1 Netzwerkarchitekturen und -protokolle

1.2 Die Internet-Protokoll-Suite

1.3 Datenkapselung

1.3.1 Header, Footer und Adressen

1.3.2 Datenübertragung

1.4 Netzwerk-Routing

1.5 Mein Modell für die Analyse von Netzwerkprotokollen

1.6 Am Ende dieses Kapitels

## **2 Capturing von Anwendungsverkehr**

2.1 Passives Capturing von Netzwerkverkehr

2.2 Eine kurze Einführung in Wireshark

2.3 Alternative passive Capturing-Techniken

2.3.1 Tracing von Systemaufrufen

2.3.2 Das strace-Utility unter Linux

2.3.3 Netzwerkverbindungen mit DTrace verfolgen

2.3.4 Process Monitor unter Windows

2.4 Vor- und Nachteile passiven Capturings

2.5 Aktives Capturing von Netzwerkverkehr

2.6 Netzwerk-Proxys

2.6.1 Port-Forwarding-Proxy

2.6.2 SOCKS-Proxy

2.6.3 HTTP-Proxys

2.6.4 Forwarding eines HTTP-Proxys

2.6.5 HTTP-Reverse-Proxy

2.7 Am Ende dieses Kapitels

### **3 Strukturen von Netzwerk-Protokollen**

3.1 Binäre Protokollstrukturen

3.1.1 Numerische Daten

3.1.2 Boolesche Werte

3.1.3 Bit-Flags

3.1.4 Binäre Bytereihenfolge (Endianness)

3.1.5 Text und menschenlesbare Daten

3.1.6 Binärdaten variabler Länge

3.2 Datum und Uhrzeit

3.2.1 POSIX/Unix-Zeit

3.2.2 Windows FILETIME

3.3 TLV-Muster

3.4 Multiplexing und Fragmentierung

3.5 Netzwerk-Adressinformationen

3.6 Strukturierte Binärformate

3.7 Strukturen textbasierter Protokolle

3.7.1 Numerische Daten

3.7.2 Boolesche Werte in Textform

3.7.3 Datum und Uhrzeit

3.7.4 Daten variabler Länge

3.7.5 Formate für strukturierten Text

3.8 Codierung binärer Daten

3.8.1 Hex-Codierung

3.8.2 Base64

3.9 Am Ende dieses Kapitels

## **4 Fortgeschrittenes Capturing von Anwendungsverkehr**

4.1 Rerouting von Verkehr

4.1.1 Traceroute nutzen

4.1.2 Routing-Tabellen

4.2 Konfiguration eines Routers

4.2.1 Routing unter Windows aktivieren

4.2.2 Routing unter \*nix aktivieren

4.3 Network Address Translation

4.3.1 SNAT aktivieren

4.3.2 SNAT unter Linux konfigurieren

4.3.3 DNAT aktivieren

4.4 Verkehr an ein Gateway weiterleiten

4.4.1 DHCP-Spoofing

4.4.2 ARP-Poisoning

4.5 Am Ende dieses Kapitels

## **5 Analyse auf der Datenleitung**

5.1 Die Verkehr produzierende Anwendung: SuperFunkyChat

5.1.1 Den Server starten

5.1.2 Clients starten

5.1.3 Kommunikation zwischen Clients

5.2 Ein Crashkurs zur Analyse mit Wireshark

5.2.1 Netzwerkverkehr generieren und Pakete erfassen

- 5.2.2 Grundlegende Analyse
- 5.2.3 Inhalte einer TCP-Session lesen
- 5.3 Die Paketstruktur mit Hex Dump identifizieren
  - 5.3.1 Einzelne Pakete betrachten
  - 5.3.2 Die Protokollstruktur ermitteln
  - 5.3.3 Unsere Annahmen überprüfen
  - 5.3.4 Das Protokoll mit Python sezieren
- 5.4 Einen Wireshark-Dissector in Lua entwickeln
  - 5.4.1 Den Dissector entwickeln
  - 5.4.2 Sezieren mit Lua
  - 5.4.3 Parsen eines Nachrichtenpakets
- 5.5 Einen Proxy zur aktiven Verkehrsanalyse nutzen
  - 5.5.1 Den Proxy einrichten
  - 5.5.2 Protokollanalyse mittels Proxy
  - 5.5.3 Grundlegendes Parsen von Protokollen hinzufügen
  - 5.5.4 Das Protokollverhalten ändern
- 5.6 Am Ende dieses Kapitels

## **6 Reverse Engineering einer Anwendung**

- 6.1 Compiler, Interpreter und Assembler
  - 6.1.1 Interpretierte Sprachen
  - 6.1.2 Kompilierte Sprachen
  - 6.1.3 Statisches und dynamisches Linking
- 6.2 Die x86-Architektur
  - 6.2.1 Instruction Set Architecture
  - 6.2.2 CPU-Register
  - 6.2.3 Ablaufsteuerung
- 6.3 Betriebssystem-Grundlagen
  - 6.3.1 Dateiformate für Executables

6.3.2 Abschnitte

6.3.3 Prozesse und Threads

6.3.4 Netzwerkschnittstelle des Betriebssystems

6.3.5 Application Binary Interface

6.4 Statisches Reverse Engineering

6.4.1 Kurzanleitung für die Nutzung der IDA Pro Free Edition

6.4.2 Stackvariablen und Argumente analysieren

6.4.3 Schlüsselfunktionalitäten identifizieren

6.5 Dynamisches Reverse Engineering

6.5.1 Breakpunkte setzen

6.5.2 Debugger-Fenster

6.5.3 Wo setzt man Breakpunkte?

6.6 Reverse Engineering von Managed Code

6.6.1 .NET-Anwendungen

6.6.2 ILSpy nutzen

6.6.3 Java-Anwendungen

6.6.4 Mit Verschleierungstaktiken umgehen

6.7 Reverse-Engineering-Ressourcen

6.8 Am Ende dieses Kapitels

## **7 Sicherheit von Netzwerkprotokollen**

7.1 Verschlüsselungsalgorithmen

7.1.1 Substitutionschiffre

7.1.2 XOR-Verschlüsselung

7.2 Zufallszahlengeneratoren

7.3 Symmetrische Verschlüsselung

7.3.1 Blockchiffre

7.3.2 Blockchiffre-Modi

7.3.3 Blockchiffre-Padding

7.3.4 Padding Oracle Attack

7.3.5 Stromchiffre

7.4 Asymmetrische Verschlüsselung

7.4.1 RSA-Algorithmus

7.4.2 RSA-Padding

7.4.3 Schlüsselaustausch nach Diffie-Hellman

7.5 Signaturalgorithmen

7.5.1 Kryptografische Hash-Algorithmen

7.5.2 Asymmetrische Signaturalgorithmen

7.5.3 Message Authentication Codes

7.6 Public-Key-Infrastruktur

7.6.1 X.509-Zertifikate

7.6.2 Verifikation einer Zertifikatskette

7.7 Fallbeispiel: Transport Layer Security

7.7.1 Der TLS-Handshake

7.7.2 Initiale Aushandlungen

7.7.3 Endpunkt-Authentifizierung

7.7.4 Die Verschlüsselung aufbauen

7.7.5 Sicherheitsanforderungen erfüllen

7.8 Am Ende dieses Kapitels

## **8 Implementierung des Netzwerkprotokolls**

8.1 Replay von erfasstem Netzwerkverkehr

8.1.1 Verkehr mit Netcat erfassen

8.1.2 Replay von UDP-Verkehr mittels Python

8.1.3 Unseren Analyse-Proxy wiederverwenden

8.2 Ausführbaren Code wiederverwenden

8.2.1 Code in .NET-Anwendungen wiederverwenden

8.2.2 Code in Java-Anwendungen wiederverwenden

8.2.3 Unmanaged Executables

8.3 Verschlüsselung und der Umgang mit TLS

8.3.1 Die verwendete Verschlüsselung ermitteln

8.3.2 TLS-Verkehr entschlüsseln

8.4 Am Ende dieses Kapitels

## **9 Die Hauptursachen für Sicherheitslücken**

9.1 Vulnerabilitätsklassen

9.1.1 Remote Code Execution

9.1.2 Denial-of-Service

9.1.3 Offenlegung von Informationen

9.1.4 Authentifizierung umgehen

9.1.5 Autorisierung umgehen

9.2 Verfälschung des Speichers

9.2.1 Speichersichere und speicherunsichere Programmiersprachen

9.2.2 Pufferüberlauf

9.2.3 Out-of-Bounds-Indexierung

9.2.4 Datenexpansion

9.2.5 Fehler bei der dynamischen Speicherallozierung

9.3 Voreingestellte oder festcodierte Anmeldedaten

9.4 Offenlegung von Benutzernamen

9.5 Fehlerhafter Zugriff auf Ressourcen

9.5.1 Kanonisierung

9.5.2 Fehlermeldungen mit zu viel Information

9.6 Speicherüberlastung

9.7 Massenspeicherüberlastung

9.8 CPU-Überlastung

9.8.1 Algorithmische Komplexität

9.8.2 Konfigurierbare Kryptografie

- 9.9 Formatstrings
- 9.10 Befehlsinjektion
- 9.11 SQL-Injektion
- 9.12 Zeichenersetzung bei Textcodierung
- 9.13 Am Ende dieses Kapitels

## **10 Sicherheitslücken aufspüren und ausnutzen**

- 10.1 Fuzzing
  - 10.1.1 Der einfachste Fuzzing-Test
  - 10.1.2 Mutations-Fuzzer
  - 10.1.3 Testdatensätze generieren
- 10.2 Sicherheitslücken untersuchen
  - 10.2.1 Debugging von Anwendungen
  - 10.2.2 Die Chancen erhöhen, um die Hauptursache für einen Absturz zu ermitteln
- 10.3 Gängige Sicherheitslücken ausnutzen
  - 10.3.1 Exploit von Speicherlücken
  - 10.3.2 Willkürliche Schreiboperationen
- 10.4 Shell-Code entwickeln
  - 10.4.1 Erste Schritte
  - 10.4.2 Einfache Debugging-Technik
  - 10.4.3 Systemaufrufe ausführen
  - 10.4.4 Andere Programme ausführen
  - 10.4.5 Shell-Code mit Metasploit generieren
- 10.5 Maßnahmen gegen Speicherlücken
  - 10.5.1 Data Execution Prevention
  - 10.5.2 Return-Oriented Programming
  - 10.5.3 Address Space Layout Randomization (ASLR)
  - 10.5.4 Stacküberläufe durch Canaries erkennen

## 10.6 Am Ende dieses Kapitels

### **Anhang**

#### A Toolkit für die Netzwerkprotokoll-Analyse

##### A.1 Tools zum passiven Capturing und zur Analyse von Netzwerkprotokollen

###### A.1.1 Microsoft Message Analyzer

###### A.1.2 TCPDump und LibPCAP

###### A.1.3 Wireshark

##### A.2 Aktives Netzwerk-Capturing und Analyse

###### A.2.1 Canape

###### A.2.2 Canape Core

###### A.2.3 Mallory

##### A.3 Netzwerkkonnektivität und Protokolltests

###### A.3.1 Hping

###### A.3.2 Netcat

###### A.3.3 Nmap

##### A.4 Webanwendungen testen

###### A.4.1 Burp Suite

###### A.4.2 Zed Attack Proxy (ZAP)

###### A.4.3 Mitmproxy

##### A.5 Frameworks zum Fuzzing, zur Paketgenerierung und zur Entwicklung von Exploits

###### A.5.1 American Fuzzy Lop (AFL)

###### A.5.2 Kali Linux

###### A.5.3 Metasploit-Framework

###### A.5.4 Scapy

###### A.5.5 Sulley

##### A.6 Netzwerk-Spoofing und -Umleitung

A.6.1 DNSMasq

A.6.2 Ettercap

A.7 Reverse Engineering von Executables

A.7.1 Java Decompiler (JD)

A.7.2 IDA Pro

A.7.3 Hopper

A.7.4 ILSpy

A.7.5 .NET Reflector

**Index**

# 4

## Fortgeschrittenes Capturing von Anwendungsverkehr

Üblicherweise reichen die Capturing-Techniken aus, die Sie in Kapitel 2 kennengelernt haben, doch gelegentlich stehen Sie vor schwierigen Situationen, die fortgeschrittene Capturing-Techniken erfordern. Manchmal stellt eine Embedded-Plattform die Herausforderung dar, die nur per DHCP (Dynamic Host Configuration Protocol) konfiguriert werden kann, ein anderes Mal ist es ein Netzwerk, über das Sie keine Kontrolle haben, solange Sie nicht direkt mit ihm verbunden sind.

Ein Großteil der in diesem Kapitel diskutierten fortgeschrittenen Techniken zum Erfassen von Netzwerkverkehr nutzt die existierende Netzwerkinfrastruktur und die vorhandenen Protokolle zur Umleitung des Verkehrs. Keine dieser Techniken verlangt spezielle Hardware. Sie benötigen nur Softwarepakete, die auf unterschiedlichen Betriebssystemen zu finden sind.

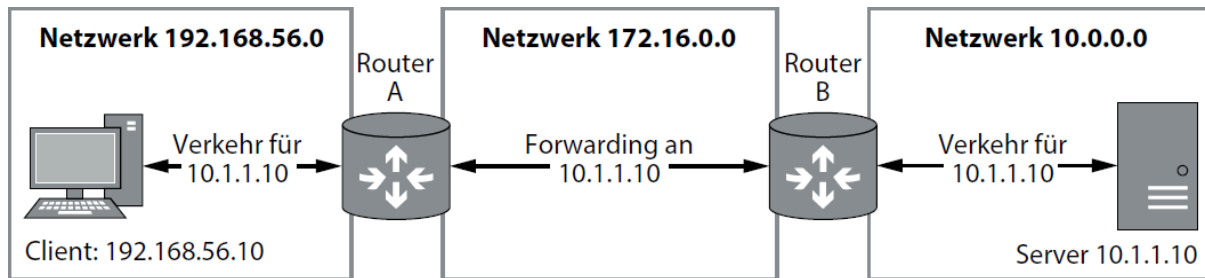
### 4.1 Rerouting von Verkehr

IP ist ein *geroutetes* Protokoll, d. h., keiner der Knoten des Netzwerks muss die genaue Lage eines anderen Knotens im Netzwerk kennen. Sendet ein Knoten Verkehr an ein anderes Netzwerk, mit dem er nicht direkt verbunden ist, dann gibt er den Verkehr an ein *Gateway* weiter, das den Verkehr an das Ziel weiterleitet. Ein Gateway wird üblicherweise auch als *Router* bezeichnet, ein Gerät, das Verkehr von einem Ort an einen anderen routet.

In Abbildung 4–1 versucht zum Beispiel der Client 192.168.56.10 Daten an den Server 10.1.1.10 zu senden, doch der Client hat keine direkte Verbindung mit dem Server. Er sendet den für den Server gedachten Verkehr daher zuerst an Router A. Router A sendet die Daten dann an Router B weiter, der eine direkte

Verbindung zum Zielserver hat. Router B leitet den Verkehr dann an sein eigentliches Ziel weiter.

Wie alle anderen Knoten des Netzwerks kennt auch das Gateway das genaue Ziel der Daten nicht und sucht sich daher ein passendes nächstes Gateway aus, an das es den Verkehr weiterleitet. In unserem Beispiel kennen die Router A und B nur die beiden Netzwerke, mit denen sie direkt verbunden sind. Um vom Client zum Server zu gelangen, muss der Verkehr geroutet werden.



**Abb. 4-1** Beispiel für gerouteten Verkehr

#### 4.1.1 Traceroute nutzen

Beim »Tracing« einer Route versuchen Sie zu verfolgen, welchen Weg der IP-Verkehr zu einem bestimmten Ziel nimmt. Die meisten Betriebssysteme besitzen fest integrierte Tools, die ein solches Tracing durchführen, z. B. `tracert` bei den meisten unixoiden Plattformen und `tracert` unter Windows.

Listing 4-1 zeigt das Ergebnis eines Tracings der Route zu `www.google.com` über eine heimische Internetverbindung.

```
C:\Users\user>tracert www.google.com
```

```
Tracing route to www.google.com [173.194.34.176]
```

```
over a maximum of 30 hops:
```

```
  1      2 ms      2 ms      2 ms  home.local [192.168.1.254]
  2     15 ms     15 ms     15 ms  217.32.146.64
  3     88 ms     15 ms     15 ms  217.32.146.110
```

4	16 ms	16 ms	15 ms	217.32.147.194
5	26 ms	15 ms	15 ms	217.41.168.79
6	16 ms	26 ms	16 ms	217.41.168.107
7	26 ms	15 ms	15 ms	109.159.249.94
8	18 ms	16 ms	15 ms	109.159.249.17
9	17 ms	28 ms	16 ms	62.6.201.173
10	17 ms	16 ms	16 ms	195.99.126.105
11	17 ms	17 ms	16 ms	209.85.252.188
12	17 ms	17 ms	17 ms	209.85.253.175
13	27 ms	17 ms	17 ms	lhr14s22-in-f16.1e100.net [173.194.34.176]

---

**Listing 4-1** *Traceroute für www.google.com mithilfe des tracert-Tools*

Jede nummerierte Ausgabezeile (1, 2 usw.) steht für ein spezifisches Gateway, das den Verkehr an das eigentliche Ziel weiterleitet. Die Ausgabe ist auf eine maximale Anzahl von *Hops* beschränkt. Ein einzelner Hop repräsentiert das Netzwerk zwischen jedem Gateway der Route. Beispielsweise gibt es einen Hop zwischen Ihrem Rechner und dem ersten Router, einen weiteren zwischen dem Router und dem nächsten Router und so weiter bis zum eigentlichen Ziel. Wird die maximale Anzahl von Hops überschritten, bricht traceroute die Suche nach weiteren Routern ab. Die maximale Zahl von Hops kann bei traceroute in der Kommandozeile festgelegt werden. Verwenden Sie `-h ANZAHL` unter Windows und `-m ANZAHL` bei unixoiden Systemen. (Die Ausgabe enthält auch die Umlaufzeit vom Startrechner der Traceroute zum gefundenen Knoten.)

## 4.1.2 Routing-Tabellen

Das Betriebssystem verwendet *Routing-Tabellen*, um zu bestimmen, an welche Gateways Verkehr geschickt werden muss. Eine Routing-Tabelle enthält eine Liste von Zielnetzwerken und Gateways, an die der Verkehr zu senden ist. Ist ein Netzwerk direkt mit dem Knoten verbunden, der die Daten sendet, dann ist kein Gateway nötig, d. h., der Verkehr kann direkt im lokalen Netzwerk übertragen werden.

Sie können sich die Routing-Tabelle Ihres Computers ansehen, indem Sie `netstat -r` bei den meisten unixoiden Systemen bzw. `route print` unter Windows eingeben. Listing 4-2 zeigt die Ausgabe dieses Befehls unter Windows.

---

```
> route print
```

```
IPv4 Route Table
```

```
=====
```

```
Active Routes:
```

Network	Destination	Interface	Metric	Netmask	Gateway
1	0.0.0.0			0.0.0.0	192.168.1.254
192.168.1.72			10		
	127.0.0.0			255.0.0.0	On-link
	127.0.0.1		306		
	127.0.0.1			255.255.255.255	On-link
	127.0.0.1		306		
	127.255.255.255			255.255.255.255	On-link
	127.0.0.1		306		

192.168.1.0	255.255.255.0	On-link
192.168.1.72	266	
192.168.1.72	255.255.255.255	On-link
192.168.1.72	266	
192.168.1.255	255.255.255.255	On-link
192.168.1.72	266	
224.0.0.0	240.0.0.0	On-link
127.0.0.1	306	
224.0.0.0	240.0.0.0	On-link
192.168.56.1	276	
224.0.0.0	240.0.0.0	On-link
192.168.1.72	266	
255.255.255.255	255.255.255.255	On-link
127.0.0.1	306	
255.255.255.255	255.255.255.255	On-link
192.168.56.1	276	
255.255.255.255	255.255.255.255	On-link
192.168.1.72	266	

=====  
=====

---

**Listing 4-2**      *Beispielhafte Routing-Tabelle*

Wie bereits erwähnt besteht ein Grund für den Einsatz von Routing darin, dass die Knoten den Ort der anderen Knoten im Netzwerk nicht kennen müssen. Doch was passiert mit Verkehr, wenn man das Gateway, das für die Kommunikation mit dem Zielnetzwerk verantwortlich ist, nicht kennt? In diesem Fall ist es üblich, dass die Routing-Tabelle den gesamten unbekanntes Verkehr an das sogenannte *Default Gateway* weiterleitet. Sie sehen dieses Standard-

Gateway bei ①, wo als Zielnetzwerk 0.0.0.0 angegeben ist. Dieses Ziel ist ein Platzhalter für das Standard-Gateway, was die Verwaltung der Routing-Tabelle vereinfacht. Durch den Platzhalter muss die Tabelle nicht geändert werden, wenn sich die Netzwerkkonfiguration ändert, etwa durch eine DHCP-Konfiguration. Verkehr, der an irgendein Ziel gerichtet ist, für das es keine passende Route gibt, wird an das Gateway mit der Platzhalteradresse 0.0.0.0 gesendet.

Wie kann man das Routing nun für seine Zwecke nutzen? Nehmen wir ein Embedded-System an, bei dem Betriebssystem und Hardware in einem einzigen Gerät vereint sind. Möglicherweise haben Sie keinen Einfluss auf die Netzwerkkonfiguration des Embedded-Systems, weil Sie gar nicht auf das zugrunde liegende Betriebssystem zugreifen können, doch wenn Sie Ihre Capturing-Vorrichtung als Gateway zwischen dem System und dem eigentlichen Ziel schalten, können Sie auch den Verkehr dieses Systems erfassen.

Die folgenden Abschnitte zeigen Möglichkeiten auf, wie man ein Betriebssystem konfiguriert, damit der Rechner als Gateway fungieren und den Netzwerkverkehr abgreifen kann.

## **4.2 Konfiguration eines Routers**

Standardmäßig leiten die meisten Betriebssysteme den Verkehr zwischen Netzwerkschnittstellen nicht direkt weiter. Das soll hauptsächlich verhindern, dass jemand auf einer Seite der Route direkt mit den Netzwerkadressen auf der anderen Seite kommunizieren kann. Ist das Routing in der Betriebssystemkonfiguration nicht aktiviert, wird jeder Verkehr, der geroutet werden müsste, verworfen (oder es gibt eine Fehlermeldung an den Sender). Die Standardkonfiguration ist für die Sicherheit sehr wichtig: Stellen Sie sich die Folgen vor, wenn der Router, der Ihre Verbindung zum Internet kontrolliert, Daten aus dem Internet direkt in Ihr privates Netzwerk leitet.

Um also das Routing im Betriebssystem zu aktivieren, müssen Sie mit Administratorrechten einige Änderungen an der Konfiguration vornehmen. Zwar ist die Aktivierung des Routings bei jedem Betriebssystem anders geregelt, doch ein Aspekt bleibt immer gleich: Der Computer muss über mindestens zwei Netzwerkschnittstellen verfügen, um als Router fungieren zu können. Darüber hinaus benötigen Sie Routen auf beiden Seiten des Gateways, damit das Routing korrekt funktioniert. Hat das Ziel keine entsprechende Route zurück zur Quelle, funktioniert die Kommunikation nicht wie gewünscht. Sobald das Routing

aktiviert ist, können Sie Ihre Geräte so konfigurieren, dass der Verkehr über Ihren neuen Router geleitet wird. Läuft ein Tool wie Wireshark auf dem Router, können Sie den gesamten Verkehr festhalten, der durch die beiden von Ihnen konfigurierten Netzwerkschnittstellen läuft.

#### 4.2.1 Routing unter Windows aktivieren

Standardmäßig ist das Routing zwischen Netzwerkschnittstellen unter Windows nicht aktiviert. Um das Routing unter Windows zu aktivieren, müssen Sie die Windows-Registrierung anpassen. Das kann über den grafischen Registrierungseditor erfolgen, doch die einfachste Lösung besteht darin, den folgenden Befehl als Administrator in der Kommandozeile auszuführen:

---

```
C> reg add
HKLM\System\CurrentControlSet\Services\Tcpip\Parameters ^

/v IPEnableRouter /t REG_DWORD /d 1
```

---

Um das Routing zu deaktivieren, nachdem Sie den gewünschten Verkehr erfasst haben, geben Sie den folgenden Befehl ein:

---

```
C> reg add
HKLM\System\CurrentControlSet\Services\Tcpip\Parameters ^

/v IPEnableRouter /t REG_DWORD /d 0
```

---

Bei Änderungen ist außerdem ein Neustart nötig.

#### **Warnung**

Sie müssen bei Änderungen an der Windows-Registrierung sehr vorsichtig sein. Fehlerhafte Änderungen können Windows komplett zerstören und das Booten verhindern! Stellen Sie sicher, dass es ein System-Backup gibt (das Sie mit dem in Windows integrierten Backup-Tool erstellen können), bevor Sie gefährliche Änderungen vornehmen.

## 4.2.2 Routing unter \*nix aktivieren

Um das Routing bei unixoiden Betriebssystemen zu aktivieren, ändern Sie einfach die Einstellung des IP-Routing-Systems mit dem Befehl `sysctl`. (Die Anweisungen sind nicht unbedingt für alle Systeme gleich, doch Sie sollten keine Schwierigkeiten haben, die entsprechenden Anweisungen zu finden.)

Um das IPv4-Routing unter Linux zu aktivieren, führen Sie den folgenden Befehl als root aus (ein Neustart ist nicht nötig; die Änderung wird sofort aktiv):

---

```
# sysctl net.ipv4.conf.all.forwarding=1
```

---

Das IPv6-Routing unter Linux aktivieren Sie wie folgt:

---

```
# sysctl net.ipv6.conf.all.forwarding=1
```

---

Sie können die Routing-Konfiguration wieder rückgängig machen, indem Sie in den obigen Befehlen die `1` durch eine `0` ersetzen.

Das Routing unter macOS aktivieren Sie wie folgt:

---

```
> sysctl -w net.inet.ip.forwarding=1
```

---

## 4.3 Network Address Translation

Beim Capturing von Netzwerkverkehr könnten Sie feststellen, dass Sie zwar ausgehenden Verkehr erfassen können, nicht aber den eingehenden Verkehr. Der Grund dafür liegt darin, dass ein Upstream-Router die Route des ursprünglichen Quellnetzwerks nicht kennt. Daher verwirft er entweder den Verkehr oder er leitet ihn an ein anderes Netzwerk weiter. Sie können dieser Situation per *Network Address Translation (NAT)* entgegenwirken. Diese Technik modifiziert die Informationen der Quell- und Zieladressen von IP und höheren Protokollen wie TCP. NAT wird ausgiebig genutzt, um den beschränkten IPv4-Adressraum zu erweitern, indem man mehrere Geräte hinter einer einzelnen öffentlichen Adresse versteckt.

NAT kann darüber hinaus die Netzwerkkonfiguration vereinfachen und sicherer machen. Bei aktiver NAT können Sie beliebig viele Geräte hinter einer einzelnen NAT-IP-Adresse betreiben und nur diese öffentliche IP-Adresse pflegen.

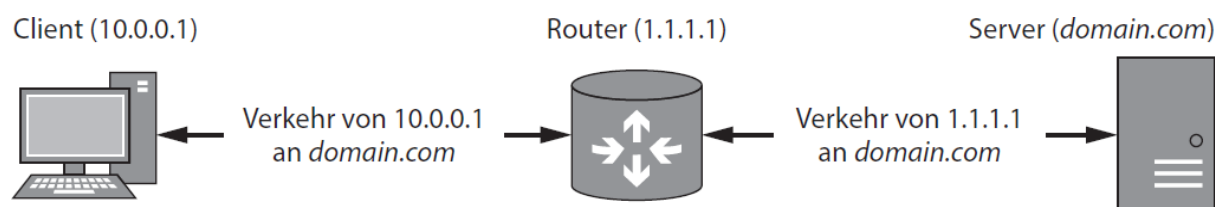
Heutzutage sind zwei Arten von NAT üblich: *Source NAT (SNAT)* und *Destination NAT (DNAT)*. Der Unterschied zwischen den beiden besteht darin, welche Adresse bei der NAT-Verarbeitung des Netzwerkverkehrs modifiziert wird. SNAT (auch *Masquerading* genannt) verändert die IP-Quelladresse, während DNAT die Zieladresse ändert.

### 4.3.1 SNAT aktivieren

Soll ein Router mehrere Rechner hinter einer einzigen IP-Adresse verstecken, verwenden Sie SNAT. Ist SNAT aktiv, wird bei Verkehr, der über die externe Netzwerkschnittstelle läuft, die Quell-IP-Adresse des Pakets so umgeschrieben, dass sie der einzelnen durch SNAT bereitgestellten IP-Adresse entspricht.

SNAT zu implementieren kann nützlich sein, wenn Sie Verkehr an ein Netzwerk routen wollen, über das Sie keine Kontrolle haben. Wie Sie wissen, müssen beide Knoten im Netzwerk korrekte Routing-Informationen besitzen, damit der Verkehr zwischen den Knoten übertragen werden kann. Ist die Routing-Information nicht korrekt, läuft der Verkehr im schlimmsten Fall nur in eine Richtung. Doch auch im besten Fall werden Sie den Verkehr nur in einer Richtung erfassen können, weil die andere Richtung über einen alternativen Pfad geroutet wird.

SNAT löst dieses Problem, indem es die Quelladresse des Verkehrs durch eine IP-Adresse ersetzt, an die der Zielknoten die Daten senden kann – üblicherweise diejenige, die der externen Adresse des Routers zugewiesen ist. Auf diese Weise kann der Zielknoten den Verkehr zurück an den Router senden. Abbildung 4-2 zeigt ein einfaches SNAT-Beispiel.



**Abb. 4-2** SNAT von einem Client an einen Server

Will der Client ein Paket an einen Server in einem anderen Netzwerk übertragen, sendet er es an den mit SNAT konfigurierten Router. Wenn der Router das Paket

vom Client empfängt, entspricht die Quelladresse der des Clients (10.0.0.1) und das Ziel ist der Server (die aus *domain.com* aufgelöste Adresse). An diesem Punkt wird SNAT genutzt: Der Router ändert die Quelladresse des Pakets in seine eigene Adresse (1.1.1.1) ab und leitet das Paket an den Server weiter.

Wenn der Server das Paket empfängt, geht er davon aus, dass es vom Router kommt, d. h., wenn er ein Paket zurückschicken will, sendet er es an 1.1.1.1. Der Router empfängt das Paket, erkennt, dass es von einer NAT-Verbindung stammt (basierend auf Zieladresse und Portnummer) und hebt die Adressänderung wieder auf, d. h., er wandelt 1.1.1.1 wieder in die ursprüngliche Clientadresse 10.0.0.1 um. Zum Schluss kann das Paket an den eigentlichen Client weitergeleitet werden, ohne dass der Server etwas über den Client oder das Routing in diesem Netzwerk weiß.

### 4.3.2 SNAT unter Linux konfigurieren

Zwar lässt sich SNAT unter Windows und macOS per Internetverbindungsfreigabe (Internet Connection Sharing) konfigurieren, ich beschreibe aber nur, wie man SNAT unter Linux einrichtet, weil es die am einfachsten zu erläuternde Plattform ist und die flexibelste Netzwerkkonfiguration bietet.

Vor der SNAT-Konfiguration müssen Sie Folgendes tun:

- Aktivieren Sie das IP-Routing wie vorhin in diesem Kapitel beschrieben.
- Ermitteln Sie den Namen der ausgehenden Netzwerkschnittstelle, für die Sie SNAT konfigurieren wollen. Sie können dazu den `ifconfig`-Befehl verwenden. Die ausgehende Schnittstelle hat einen Namen wie `eth0`.
- Notieren Sie die IP-Adresse, die an die ausgehende Schnittstelle gebunden ist und von `ifconfig` zurückgegeben wird.

Nun können Sie die NAT-Regeln mit `iptables` definieren. (Der `iptables`-Befehl ist bei Ihrer Linux-Distribution sehr wahrscheinlich bereits installiert.) Zuerst löschen Sie alle vorhandenen NAT-Regeln aus `iptables`, indem Sie den folgenden Befehl mit Root-Rechten ausführen:

---

```
# iptables -t nat -F
```

---

Hat die ausgehende Netzwerkschnittstelle eine feste Adresse, führen Sie den folgenden Befehl als root aus, um SNAT zu aktivieren. Ersetzen Sie *INTNAME* durch den Namen der ausgehenden Schnittstelle und *INTIP* mit der dieser Schnittstelle zugewiesenen IP-Adresse.

---

```
# iptables -t nat -A POSTROUTING -o INTNAME -j SNAT -to  
INTIP
```

---

Wird die IP-Adresse hingegen dynamisch zugewiesen (etwa über DHCP oder über eine Wählverbindung), verwenden Sie den folgenden Befehl, um die ausgehende IP-Adresse automatisch zu ermitteln:

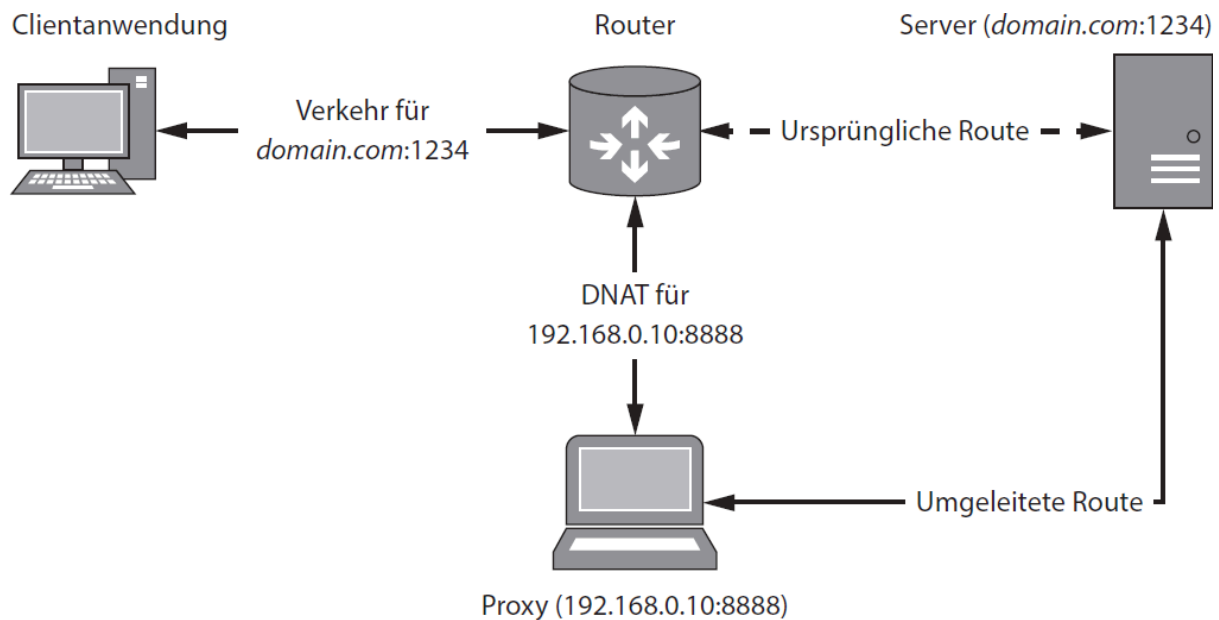
---

```
# iptables -t nat -A POSTROUTING -o INTNAME -j MASQUERADE
```

---

### 4.3.3 DNAT aktivieren

DNAT ist nützlich, wenn Sie den Verkehr an einen Proxy oder einen anderen Dienst umleiten wollen oder bevor Sie den Verkehr an sein eigentliches Ziel weiterleiten. DNAT ändert die IP-Adresse des Ziels und optional auch den Zielport. Sie können DNAT nutzen, um spezifischen Verkehr an ein anderes Ziel umzuleiten. Abbildung 4-3 zeigt, wie man Verkehr sowohl vom Router als auch vom Server auf einen Proxy an 192.168.0.10 umleitet, um eine Man-in-the-Middle-Analyse durchzuführen.



**Abb. 4-3** DNAT für einen Proxy

Abbildung 4-3 zeigt eine Clientanwendung, die den Verkehr über einen Router an *domain.com*, Port 1234 sendet. Wird ein Paket vom Router empfangen, würde er es normalerweise einfach an das Ziel weiterleiten. Da DNAT aber die Zieladresse und den Port in 192.168.0.10:8888 ändert, wendet der Router seine Weiterleitungsregeln an und sendet das Paket an den Proxy-Rechner, der den Verkehr erfassen kann. Der Proxy baut dann eine neue Verbindung zum Server auf und leitet das vom Client kommende Paket an den Server weiter. Der gesamte Verkehr zwischen dem Client und dem Server kann erfasst und manipuliert werden.

Die DNAT-Konfiguration hängt vom Betriebssystem ab, auf dem der Router läuft. (Unter Windows haben Sie wahrscheinlich kein Glück, weil die für die DNAT-Unterstützung benötigte Funktionalität dem Benutzer nicht zur Verfügung gestellt wird.) Das Setup variiert bei den verschiedenen unixoiden Betriebssystemen und macOS deutlich, weshalb ich nur die DNAT-Konfiguration unter Linux behandle. Zuerst löschen Sie alle existierenden NAT-Regeln durch Eingabe des folgenden Befehls:

---

```
# iptables -t nat -F
```

---

Als Nächstes führen Sie den folgenden Befehl als root aus, wobei Sie *ORIGIP* durch die IP-Adresse ersetzen, deren Verkehr Sie umleiten wollen, und *NEWIP* durch die neue Zieladresse, an die der Verkehr geleitet werden soll.

---

```
# iptables -t nat -A PREROUTING -d ORIGIP -j DNAT --to-destination NEWIP
```

---

Die neue NAT-Regel leitet jedes an *ORIGIP* gerichtete Paket an *NEWIP* um. (Da die DNAT unter Linux vor den normalen Routing-Regeln erfolgt, können Sie bedenkenlos eine lokale Netzwerkadresse angeben. Die DNAT-Regel hat keinen Einfluss auf Verkehr, der direkt von Linux gesendet wird.) Um die Regel nur auf TCP oder UDP anzuwenden, ändern Sie den Befehl wie folgt:

---

```
iptables -t nat -A PREROUTING -p PROTO -d ORIGIP --dport ORIGPORT -j DNAT \

--to-destination NEWIP:NEWPORT
```

---

Der Platzhalter *PROTO* (für Protokoll) muss `tcp` oder `udp` lauten, je nachdem, welches IP-Protokoll Sie mit der DNAT-Regel umleiten wollen. Die Werte für *ORIGIP* und *NEWIP* bleiben gleich.

Sie können auch *ORIGPORT* und *NEWPORT* konfigurieren, wenn Sie den Zielport ändern wollen. Wird *NEWPORT* nicht angegeben, ändert sich nur die IP-Adresse.

#### **4.4 Verkehr an ein Gateway weiterleiten**

Sie haben Ihr Gateway so eingerichtet, dass es den Verkehr erfassen und modifizieren kann. Alles scheint zu funktionieren, doch es gibt ein Problem: Sie können nicht einfach so die Netzwerkkonfiguration des Gerätes ändern, dessen Daten Sie abgreifen wollen. Auch sind Ihre Möglichkeiten beschränkt, die Netzwerkkonfiguration der Einheit zu ändern, mit dem dieses Gerät verbunden ist. Sie müssen das sendende Gerät also irgendwie dazu bringen, Verkehr über Ihr Gateway zu leiten. Sie erreichen das durch Paket-Spoofing von DHCP oder ARP (*Address Resolution Protocol*) im lokalen Netzwerk.

### 4.4.1 DHCP-Spoofing

DHCP wurde entworfen, um die Knoten in einem IP-Netzwerk automatisch mit Konfigurationsinformationen zu versorgen. Wenn Sie also den DHCP-Verkehr fälschen können, können Sie die Netzwerkkonfiguration eines Knotens aus der Ferne ändern. Wird DHCP genutzt, kann die an den Knoten gesendete Netzwerkkonfiguration eine IP-Adresse und ein Standard-Gateway umfassen, aber auch Routing-Tabellen, den Standard-DNS-Server und weitere Parameter. Verwendet das zu testende Gerät DHCP zur Konfiguration der Netzwerkschnittstelle, macht es diese Flexibilität sehr leicht, eine eigene Konfiguration zu senden, die ein einfaches Abgreifen des Verkehrs erlaubt.

DHCP nutzt das UDP-Protokoll, um Requests von und an einen DHCP-Service im lokalen Netzwerk zu übertragen. Vier Arten von DHCP-Paketen werden gesendet, um die Netzwerkkonfiguration auszuhandeln:

- **Discover**

Wird an alle Knoten des IP-Netzwerks gesendet, um einen DHCP-Server aufzuspüren.

- **Offer**

Wird vom DHCP-Server an den Knoten geschickt, der das Discovery-Paket gesendet hat, um eine Netzwerkkonfiguration anzubieten.

- **Request**

Wird vom ursprünglichen Knoten gesendet, um das Angebot (Offer) anzunehmen.

- **Acknowledgment**

Wird vom Server gesendet, um den Abschluss der Konfiguration zu bestätigen.

Interessant an DHCP ist, dass es ein authentifizierungs- und verbindungsfreies Protokoll für die Konfiguration verwendet. Selbst wenn ein DHCP-Server im Netzwerk existiert, sollten Sie den Konfigurationsprozess fälschen und die Netzwerkkonfiguration des Knotens ändern können, einschließlich der Adresse des Standard-Gateways (das Sie kontrollieren). Diese Technik bezeichnet man als *DHCP-Spoofing*.

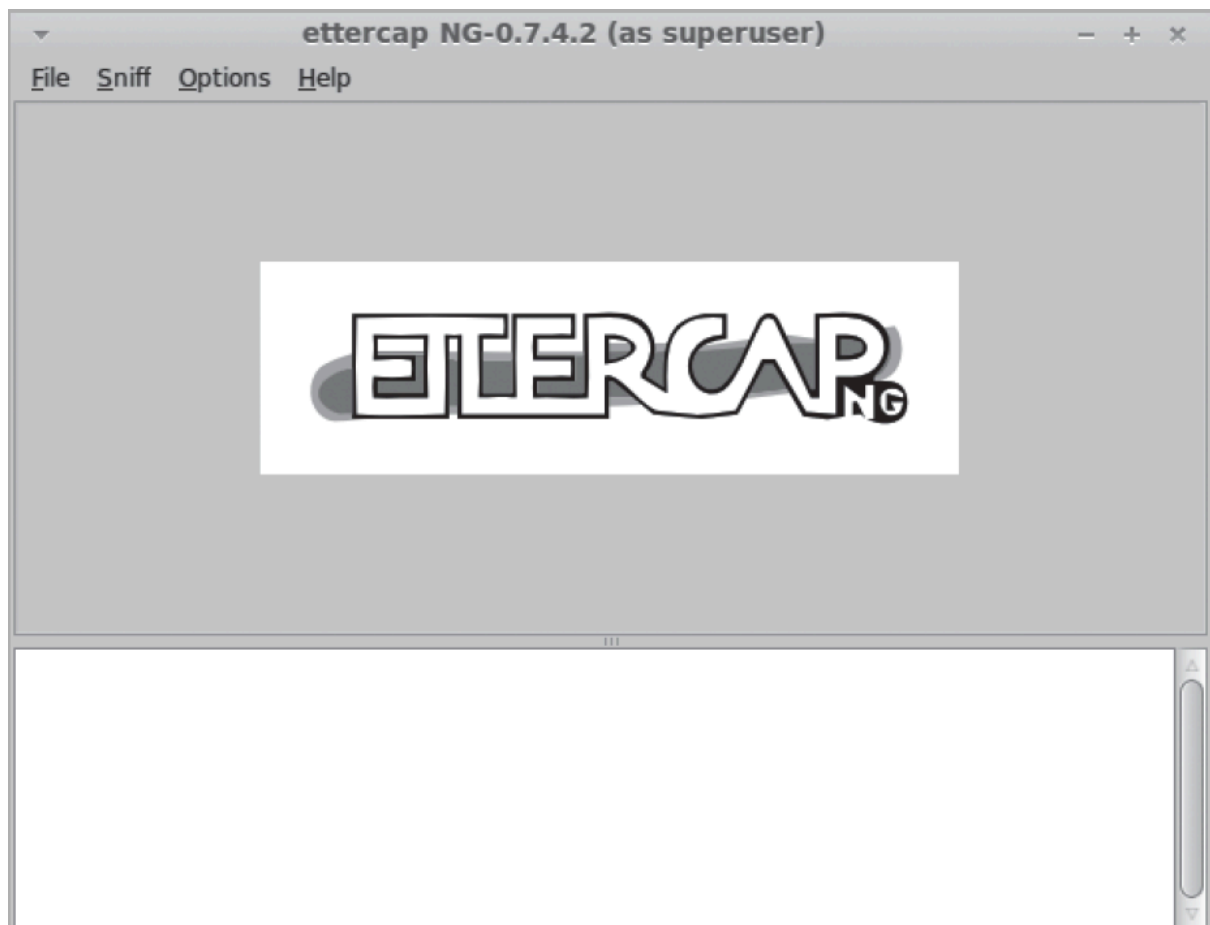
Um das DHCP-Spoofing durchzuführen, verwenden wir *Ettercap*, ein freies Tool, das für die meisten Betriebssysteme verfügbar ist (auch wenn Windows

offiziell nicht unterstützt wird).

1. Unter Linux starten Sie Ettercap im grafischen Modus als root:

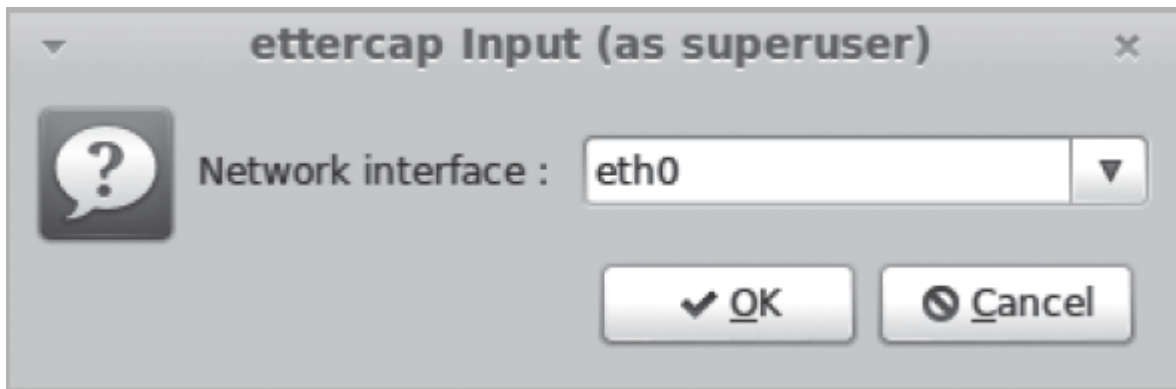
```
# ettercap -G
```

Es erscheint die Ettercap-GUI wie in Abbildung 4-4.



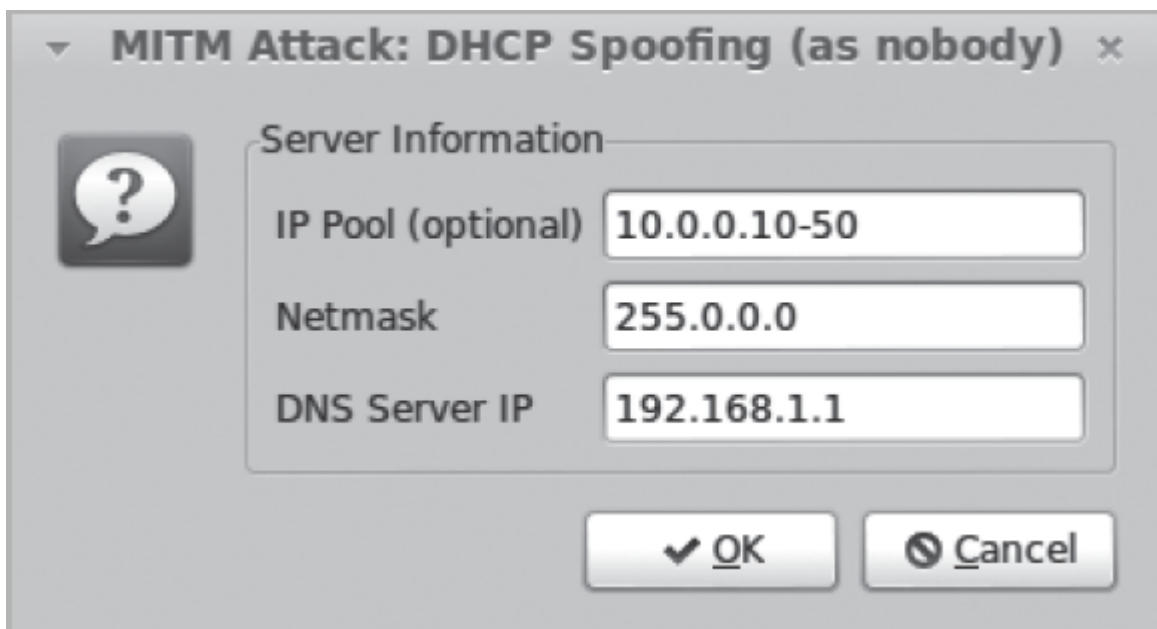
**Abb. 4-4** Die Haupt-GUI von Ettercap

2. Konfigurieren Sie Ettercaps Sniffing-Modus über **Sniff ► Unified Sniffing**.
3. Der Dialog in Abbildung 4-5 fordert Sie auf, die Netzwerkschnittstelle für das Sniffing auszuwählen. Wählen Sie diejenige aus, an der das DHCP-Spoofing erfolgen soll. (Achten Sie auf die korrekte Konfiguration des Netzwerks der Schnittstelle, da Ettercap automatisch die konfigurierte IP-Adresse der Schnittstelle als DHCP-Standard-Gateway sendet.)



**Abb. 4-5** Sniffing-Interface wählen

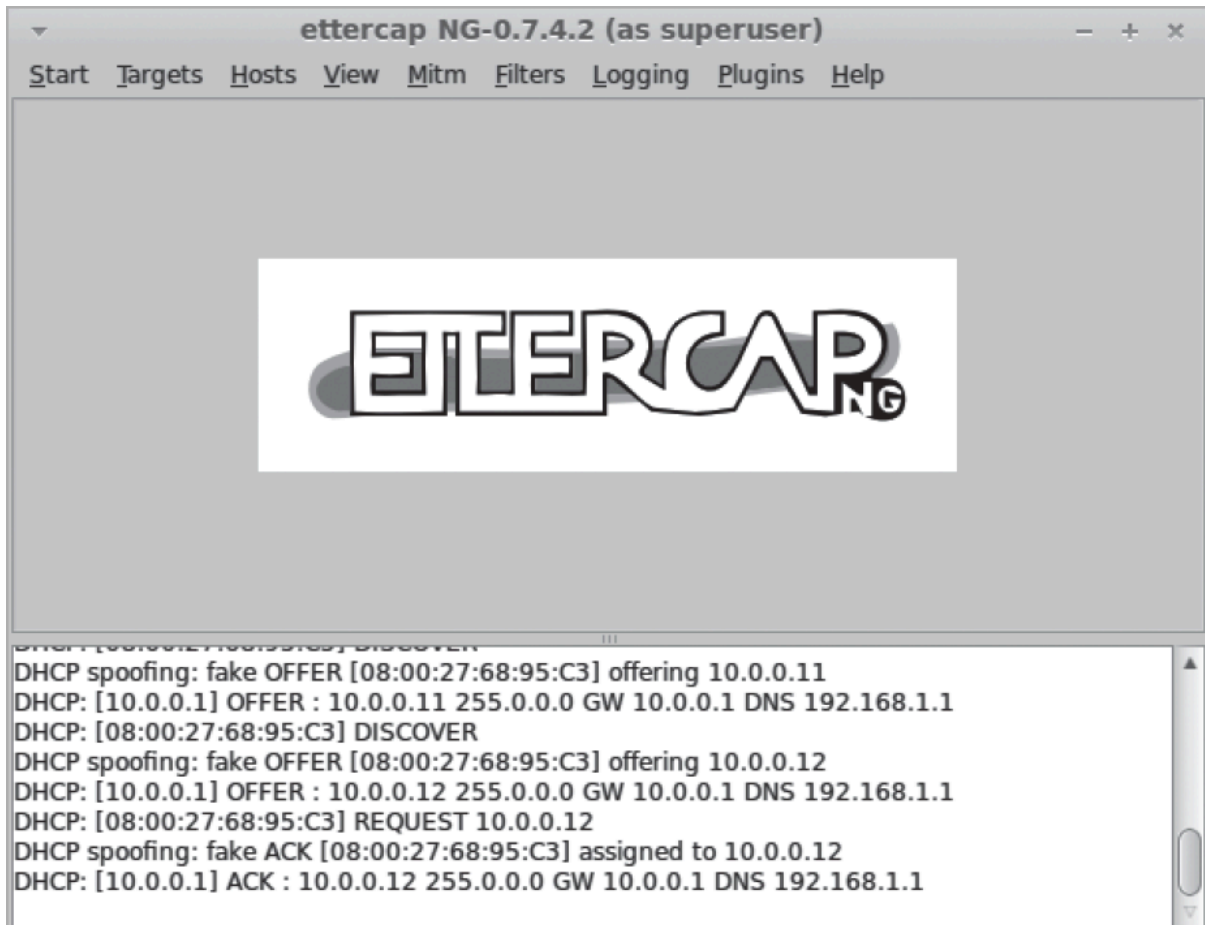
- Aktivieren Sie das DHCP-Spoofing über **MITM** ▶ **DHCP Spoofing**. Der Dialog aus Abbildung 4-6 erscheint, über den Sie die DHCP-Spoofing-Optionen einstellen können.



**Abb. 4-6** DHCP-Spoofing konfigurieren

- Das Feld IP-Pool legt den Bereich von IP-Adressen fest, die bei DHCP-Requests zurückgegeben werden. Tragen Sie den IP-Adressbereich ein, den Sie für die Netzwerkschnittstelle konfiguriert haben, die den Verkehr erfasst. In Abbildung 4-6 ist der Wert für den IP-Pool z. B. auf 10.0.0.10-50 gesetzt (das Minuszeichen legt alle Adressen innerhalb dieser Werte fest), d. h., wir geben IP-Adressen von 10.0.0.10 bis 10.0.0.50 (einschließlich) zurück. Stellen Sie die Netzmaske entsprechend der Netzmaske Ihres Netzwerks ein, um Konflikte zu vermeiden. Tragen Sie einen DNS-Server Ihrer Wahl ein.

6. Starten Sie das Sniffing über **Start ▶ Start sniffing**. Ist das DHCP-Spoofing bei einem Gerät erfolgreich, sollte das Ettercap-Logfenster wie in Abbildung 4–7 aussehen. Die maßgebliche Zeile ist das fake ACK, das von Ettercap als Response auf den DHCP-Request gesendet wurde.



**Abb. 4-7** Erfolgreiches DHCP-Spoofing

Mehr gibt es beim DHCP-Spoofing mit Ettercap nicht zu tun. Es kann eine sehr leistungsfähige Lösung sein, wenn Sie keine andere Möglichkeit haben und wenn ein DHCP-Server im angegriffenen Netzwerk läuft.

#### 4.4.2 ARP-Poisoning

ARP (Address Resolution Protocol) ist für den Betrieb von über Ethernet laufenden IP-Netzwerken sehr wichtig, weil ARP die Ethernet-Adresse für eine gegebene IP-Adresse ermittelt. Ohne ARP wäre es sehr schwierig, IP-Verkehr effizient über Ethernet zu übertragen. ARP funktioniert wie folgt: Möchte ein Knoten im gleichen Ethernet-Netzwerk mit einem anderen Knoten kommunizieren, muss er die IP-Adresse auf eine Ethernet-MAC-Adresse abbilden können. Erst durch die MAC-Adresse kennt Ethernet den Zielknoten, an den die

Daten gesendet werden sollen. Der Knoten erzeugt ein ARP-Request-Paket (siehe Abb. 4–8), die die 6-Byte-MAC-Adresse des Knotens, seine aktuelle IP-Adresse und die IP-Adresse des Zielknotens enthält. Das Paket wird im Ethernet an die MAC-Zieladresse ff:ff:ff:ff:ff:ff gesendet, was der definierten Broadcast-Adresse entspricht. Normalerweise verarbeitet eine Ethernet-Schnittstelle nur Pakete, deren Zieladresse mit ihrer eigenen Adresse übereinstimmt, doch wenn die Zieladresse der Broadcast-Adresse entspricht, wird das Paket ebenfalls verarbeitet.

Hat einer der Empfänger dieser Broadcast-Nachricht diese Ziel-IP-Adresse, kann er eine entsprechende ARP-Response zurückgeben (siehe Abb. 4–9). Diese Response ist nahezu identisch, nur dass die Felder für Sender und Empfänger vertauscht sind. Weil die IP-Adresse des Senders der ursprünglich angeforderten Ziel-IP-Adresse entsprechen sollte, kann die anfordernde Seite die MAC-Adresse des Senders speichern und für die zukünftige Netzwerkkommunikation nutzen, ohne den ARP-Request erneut senden zu müssen.

```

⊕ Frame 261: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
⊕ Ethernet II, Src: CadmusCo_01:62:d7 (08:00:27:01:62:d7), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⊖ Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IP (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: CadmusCo_01:62:d7 (08:00:27:01:62:d7)
  Sender IP address: 192.168.56.101 (192.168.56.101)
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 192.168.56.1 (192.168.56.1)

```

**Abb. 4–8** Beispiel für ein ARP-Request-Paket

```

⊕ Frame 262: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
⊕ Ethernet II, Src: CadmusCo_00:f4:8b (08:00:27:00:f4:8b), Dst: CadmusCo_01:62:d7 (08:00:27:01:62:d7)
⊖ Address Resolution Protocol (reply)
  Hardware type: Ethernet (1)
  Protocol type: IP (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (2)
  Sender MAC address: CadmusCo_00:f4:8b (08:00:27:00:f4:8b)
  Sender IP address: 192.168.56.1 (192.168.56.1)
  Target MAC address: CadmusCo_01:62:d7 (08:00:27:01:62:d7)
  Target IP address: 192.168.56.101 (192.168.56.101)

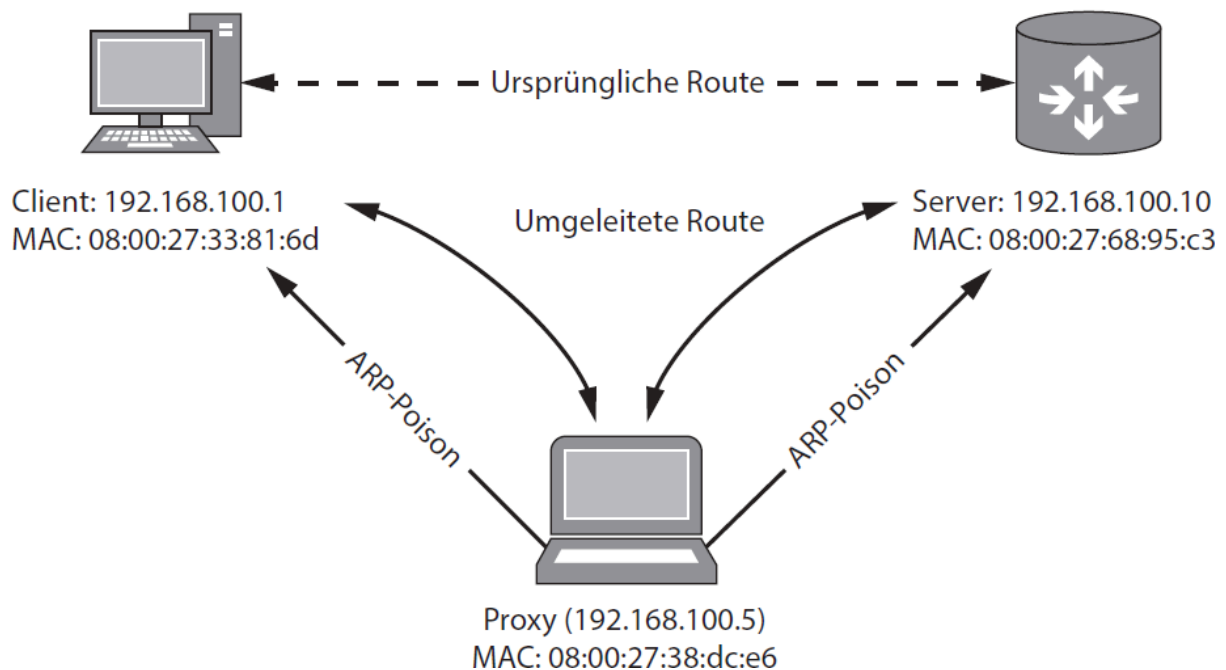
```

**Abb. 4–9** Beispiel für eine ARP-Response

Wie kann man ARP-Poisoning nun zum eigenen Vorteil nutzen? Wie bei DHCP gibt es keine Authentifizierung für ARP-Pakete, die bewusst an alle Knoten des Ethernet-Netzwerks gesendet werden. Sie können den Zielknoten also darüber informieren, dass Sie eine bestimmte IP-Adresse haben und sicherstellen, dass der Knoten den Verkehr an Ihr falsches Gateway weiterleitet, indem Sie gefälschte ARP-Pakete senden, um den ARP-Cache des Zielknotens zu

»vergiften« (engl. poisoning). Zum Fälschen der Pakete können Sie Ettercap verwenden, wie in Abbildung 4–10 zu sehen.

#### Netzwerk 192.168.100.0



**Abb. 4-10** ARP-Poisoning

In Abbildung 4–10 sendet Ettercap im lokalen Netzwerk gefälschte ARP-Pakete an den Client und an den Router. Ist das Spoofing erfolgreich, ändern diese ARP-Pakete den ARP-Cache beider Rechner so ab, dass die Einträge auf Ihren Proxy verweisen.

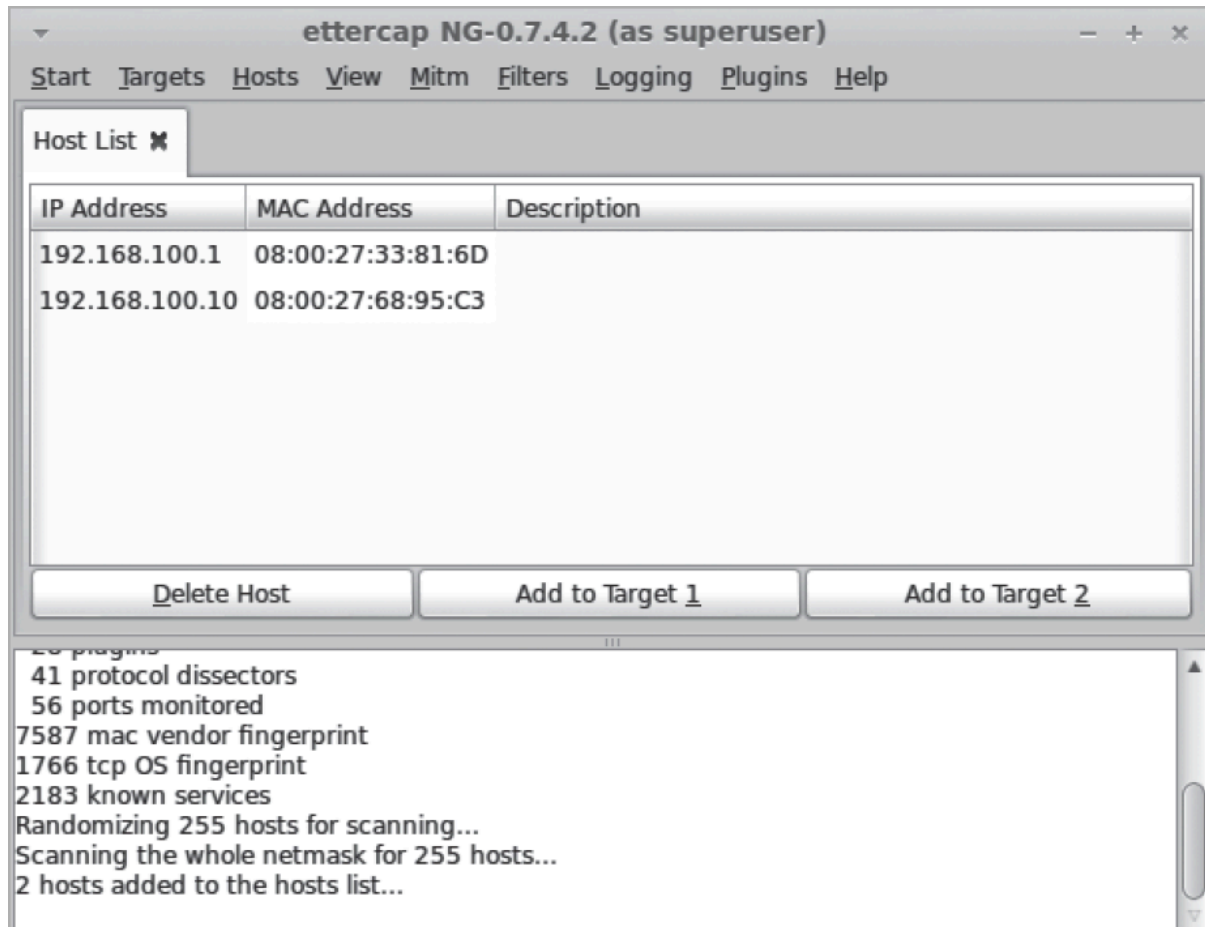
#### **Warnung**

Stellen Sie sicher, dass die ARP-Pakete sowohl für den Client als auch für den Router gefälscht werden, damit Sie beide Seiten der Kommunikation erfassen können. Soll nur eine Seite der Kommunikation abgefangen werden, müssen Sie natürlich nur eine der beiden Knoten »vergiften«.

Das ARP-Poisoning starten Sie mit den folgenden Schritten:

1. Starten Sie Ettercap und wechseln Sie wie beim DHCP-Spoofing in den **Unified Sniffing**-Modus.
2. Wählen Sie die Netzwerkschnittstelle aus, die Sie angreifen wollen (diejenige, die mit dem Netzwerk verbunden ist, dessen Knoten Sie angreifen wollen).
3. Konfigurieren Sie eine Liste von Hosts, auf die das ARP-Poisoning angewandt werden soll. Die einfachste Möglichkeit, eine solche Liste zu

erzeugen, besteht darin, sie durch Ettercap über **Hosts ▶ Scan For Hosts** scannen zu lassen. Je nach Größe des Netzwerks kann der Scan ein paar Sekunden oder auch Stunden dauern. Ist der Scan abgeschlossen, wählen Sie **Hosts ▶ Host List**. Ein Dialog wie in Abbildung 4–11 erscheint.



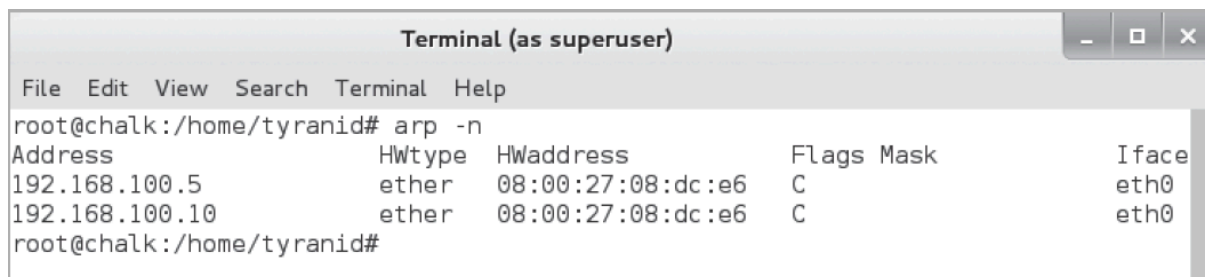
**Abb. 4–11** Liste entdeckter Hosts

Wie Sie in Abbildung 4–11 sehen, wurden zwei Hosts gefunden. In diesem Beispiel ist einer der Client, dessen Daten Sie abgreifen wollen. Er hat die IP-Adresse 192.168.100.1 und die MAC-Adresse 08:00:27:33:81:6d. Der andere Knoten ist das Gateway zum Internet mit der IP-Adresse 192.168.100.10 und der MAC-Adresse 08:00:27:68:95:c3. Wahrscheinlich kennen Sie die IP-Adressen der Netzwerkgeräte bereits, d. h., Sie können einfach feststellen, welches die lokale und welches die entfernte Maschine ist.

4. Wählen Sie Ihre Ziele. Wählen Sie einen der Hosts aus der Liste und klicken Sie auf **Add to Target 1**. Wählen Sie den anderen anzugreifenden Host und klicken Sie auf **Add to Target 2**. (Target 1 und Target 2 grenzen

Client und Gateway ab.) Das sollte das ARP-Poisoning in einer Richtung aktivieren, bei dem Daten von Target 1 an Target 2 umgeleitet wird.

5. Starten Sie das ARP-Poisoning über **MITM ▶ ARP poisoning**. Ein Dialog erscheint. Akzeptieren Sie die Voreinstellungen und klicken Sie auf **OK**. Ettercap versucht nun, den ARP-Cache des gewählten Ziels zu fälschen. Eventuell wird das ARP-Poisoning nicht sofort funktionieren, weil der ARP-Cache noch aktualisiert werden muss. Ist das Poisoning erfolgreich, sieht der Clientknoten etwa so aus, wie in Abbildung 4–12.



```
Terminal (as superuser)
File Edit View Search Terminal Help
root@chalk:/home/tyranid# arp -n
Address          HWtype  HWaddress          Flags Mask          Iface
192.168.100.5    ether   08:00:27:08:dc:e6  C                   eth0
192.168.100.10  ether   08:00:27:08:dc:e6  C                   eth0
root@chalk:/home/tyranid#
```

**Abb. 4–12** Erfolgreiches ARP-Poisoning

Abbildung 4–12 zeigt, dass der Router an IP 192.168.100.10 vergiftet wurde. Seine MAC-Hardwareadresse wurde in die MAC-Adresse des Proxys 08:00:27:08:dc:e6 geändert. (Sehen Sie sich zum Vergleich den entsprechenden Eintrag aus Abb. 4–11 an.) Jeder vom Client an den Router gesendete Verkehr läuft nun über den Proxy (zu erkennen an der MAC-Adresse von 192.168.100.5). Der Proxy kann den Verkehr an das eigentliche Ziel weiterleiten, nachdem er ihn erfasst oder modifiziert hat.

Ein Vorteil von ARP-Poisoning gegenüber DHCP-Spoofing besteht darin, dass Sie Knoten im lokalen Netzwerk dazu bringen können, mit Ihrem Gateway zu kommunizieren, selbst wenn das Ziel im lokalen Netzwerk liegt. ARP-Poisoning muss die Verbindung zwischen dem Knoten und dem externen Gateway nicht fälschen, wenn Sie das nicht wünschen.

## 4.5 Am Ende dieses Kapitels

In diesem Kapitel haben Sie einige zusätzliche Möglichkeiten kennengelernt, den Verkehr zwischen Client und Server zu erfassen und zu modifizieren. Ich habe zuerst erläutert, wie Sie Ihr Betriebssystem als IP-Gateway konfigurieren, denn wenn Sie Verkehr durch ein eigenes Gateway leiten können, stehen Ihnen eine Vielzahl von Techniken zur Verfügung.

Ein Gerät dazu zu bewegen, Verkehr an Ihre Capturing-Vorrichtung zu senden, ist natürlich nicht immer einfach. Der Einsatz von Techniken wie DHCP-Spoofing und ARP-Poisoning ist daher wichtig, um sicherzustellen, dass der Verkehr über Ihre Vorrichtung läuft, und nicht direkt über das Internet. Wie Sie gesehen haben, benötigen wir dazu glücklicherweise keine maßgeschneiderten Tools. Alle erforderlichen Tools sind bereits im Betriebssystem vorhanden (insbesondere unter Linux) oder können einfach heruntergeladen werden.

# Index

## A

ABI *siehe Application Binary Interface*

Ablaufsteuerung 131

Abstract Syntax Notation 1 (ASN.1) 59

Absturz

- analysieren 266

- beispielhafter 269

accept 136

ACK *siehe Acknowledgment*

Acknowledgment (ACK) 46, 78

add() 138

ADD (Instruktion) 127

Address Resolution Protocol (ARP) 7, 78, 81

- Poisoning 81, 85

Address Sanitizer (ASan) 272

Address Space Layout Randomization (ASLR) 302

Adresse 5

- relative 294

Adressierung 2

Advanced Encryption Standard (AES) 168, 170

AES *siehe Advanced Encryption Standard*

AFL *siehe American Fuzzy Lop*

- aktives Capturing 13, 23
- aktives Netzwerk-Capturing 315
- algorithmische Komplexität 252
- Algorithmus
  - Analyse kryptografischer Algorithmen 150
  - DH~ 183
  - Digital Signature Algorithm (DSA) 186
  - Key-Scheduling~ 169
  - kryptografischer 149
    - Hash~ 184
  - MD5 148
    - Hash~ 148, 187
      - Blockstruktur 188
  - Message Digest Algorithm (MDA) 185
  - RSA~ 180
  - Secure Hashing Algorithm (SHA) 185
  - SHA-1 (Hash-Algorithmus) 187
  - Signatur~ 184
    - asymmetrischer 186
  - Verschlüsselungs~ 148, 164
- Allozierung in definierten Speicherpools 282
- alternative Modi 174
- AMD 126
- American Fuzzy Lop (AFL) 320
- Analyse
  - eines Typs durch ILSpy 157
  - grundlegende 93
  - Häufigkeits~ 165

- kryptografischer Algorithmen 150
- Krypto~ 164
- Protokollanalyse mittels Proxy 117
- Verkehrs~ 114
- von Abstürzen 266
- von Argumenten 143
- von Netzwerkprotokollen 311
- von Stackvariablen 143
- von Strings 148
- AND (Instruktion) 127
- Antivirenprodukte 27
- Anwendung 4
- Anwendungsschicht 3–4
- Application Binary Interface (ABI) 137
- apt 36
- Argument analysieren 143
- ARM (Prozessorarchitektur) 47, 131
- ARP *siehe Address Resolution Protocol*
- ASan *siehe Address Sanitizer*
- ASCII 258
  - Standard 47
  - Zeichen 48
- ASLR *siehe Address Space Layout Randomization (ASLR)*
- ASN.1 *siehe Abstract Syntax Notation 1*
- Assembler 125
  - Sprache 125
- Assembly 154
  - laden 213

- Namen 213
- asymmetrische Verschlüsselung 179
- asymmetrischer Signaturalgorithmus 186
- Attribut 65
- AT&T-Syntax 128
- Auffüllmuster 174
- Aufspüren von Sicherheitsproblemen 261
- ausgehender Verkehr 98
- Authentifizierung 163
  - Endpunkt-~ 195
- Authentifizierungslücke 235
- Autorisierung umgehen 235
- Autorisierungslücke 235
- B**
- Backslash (\) 52
- Base Class Library (BCL) 154, 158
- Base64 66
  - Codierungstabelle 67
- Basic Constraint 192
- Basisklassenbibliothek *siehe Base Class Library (BCL)*
- BCL *siehe Base Class Library*
- bedingte Verzweigung 131
- Befehlsinjektion 256
- beispielhafter Absturz 269
- Benutzername, Offenlegung 245
- Benutzerschnittstelle 5
  - Haupt-~ 140
  - ILSpy 155

- Berkeley Packet Filter (BPF) 202
  - Ausdruck 202
- Berkeley Software Distribution (BSD) 17, 134
- Berkeley-Sockets-Modell 17, 134
- Bestätigungs-Flag ACK 46
- Betriebsmodus 172
- Bibliothek
  - Base Class Library (BCL) 154, 158
  - Canape-Core-~ 24
  - Crypt32.dll 147
  - ctypes-~ 219
  - C-~ 300
  - dynamische 220
  - mit Python laden 221
    - kompliziertere Funktionen aufrufen 222
  - OpenSSL 147
  - struct-~ 100
  - Winsock-~ 134
  - ZLIB 148
- Big Endian 46
- Big-O-Notation
  - Berechnungskomplexität 253
- Binärdatei 101
- binäre Daten
  - Codierung 65
  - Format 42
- Binary
  - dotnet-~ 88

- mono~ 88
- bind 18
- Binding-Flag 216
- Bit-Flag 46
- Bit-Flipper 263
- Bitformat 42
- Blockchiffre 168, 171
  - Modi 171
  - Padding 174
- Blowfish 171
- boolesche Werte 45, 61
- BPF *siehe Berkeley Packet Filter*
- Breakpunkt 151, 289
  - setzen 289
- Bridging-Klasse 218
- Brute-Force-Angriff 168
- BSD *siehe Berkeley Software Distribution*
- bss 133
- Bubble Sort 252
- Bucket 253
- Burp Suite 318
- Byte 42
  - Ordnung 100
  - Reihenfolge *siehe Endianness*
- C**
- CA *siehe Certificate Authority*
- CALL (Instruktion) 127
- Camellia 171

- Canape 315
- Canape Core 316
  - Bibliothek 24
- Canaries 305
  - durch einen Stackunterlauf umgehen 307
  - durch Manipulation lokaler Variablen umgehen 306
- capture.pcap 202
- Capturing
  - aktives 13, 23
  - mit Wireshark 93
  - passives 13, 22, 311
  - Techniken 22
    - passive 16
  - von Beispielverkehr 206
- C-Äquivalent, natives 222
- Carriage Return (CR) 62
- ca.crt-Datei 228
- ca.pfx-Datei 228
- CBC *siehe Cipher Block Chaining*
- C-Bibliothek 300
- C-Bibliotheksfunktion rand() 168
- CDB *siehe Console Debugger*
- cdecl 224
- cdll 221, 224
- Cert
  - Issuer 225
  - Subject 225
- Certificate Authority (CA) 191

- Certificate Revocation List (CRL) 192
- certmgr.msc 228
- change cipher spec-Paket 198
- ChatClient 87, 89
- ChatServer 87
  - einfacher 210
- ChatServer.exe 88
- Chiffretext 164
- Chinesisch, Japanisch, Koreanisch (CJK) 49
- Chosen Plaintext Attack 182
- CIL *siehe Common Intermediate Language*
- Cipher 164
- Cipher Block Chaining (CBC) 172
  - Betriebsmodus 173
- CJK *siehe Chinesisch, Japanisch, Koreanisch*
- Client
  - einfachen Netzwerkclient implementieren 207
  - Kommunikation zwischen Clients 89
  - starten 88
  - Verbindung mit einem TCP-Server herstellen 136
  - Zertifikat 197
- CLR *siehe Common Language Runtime*
- CMP (Instruktion) 127
- Code 164
  - Fehler~ 293
  - in Java-Anwendungen wiederverwenden 217
  - in .NET-Anwendungen wiederverwenden 212
  - Punkt 49

- Seite 48
- Shell~ 286, 290
  - Metasploit 296
  - wiederverwenden 211
- Codepunkt 49
- Codierung
  - Base64 66
    - Tabelle 67
  - binärer Daten 65
  - Hex~ 66
  - Prozent~ 66
  - Unicode~ 50
  - von Daten 2
  - Zeichen~ 49
- Codierungsschicht 9
- Common Intermediate Language (CIL) 153
- Common Language Runtime (CLR) 154
- Common Object Request Broker Architecture (CORBA) 26
- Compiler 125, 221
- connect 18
- Connect ()-Methode
  - öffentliche 215
  - private 215
- Console Debugger (CDB) 264
- Content Layer *siehe Inhaltsschicht*
- Cookies 305
- CORBA *siehe Common Object Request Broker Architecture*
- CPU

Register 128  
Überlastung 252  
CR *siehe Carriage Return*  
CRC *siehe Cyclic Redundancy Check*  
CRL *siehe Certificate Revocation List*  
cron-Job 284  
Cross-Site Scripting (XSS) 64  
CryptoAllPermissionCollection.class 159  
Crypt32.dll (Verschlüsselungsbibliothek) 147  
CS (Selektorregister) 130  
ctypes-Bibliothek 219  
curl 36  
Cyclic Redundancy Check (CRC) 185

**D**

Dante 31  
data 133  
Data Encryption Standard (DES) 168–169  
    Cracker 170  
Data Execution Prevention (DEP) 298  
Datagram Transport Layer Security (DTLS) 193  
Datagramm 6  
    Socket 135  
Datei capture.pcap 202  
Dateiformat für Executables 132  
Daten  
    Codierung 2  
        binärer Daten 65  
    Expansion 243

- Expansionsangriff 244
- festcodierte Anmelde~ 244
- Formatierung 2
- garantierte Reihenfolge von ~ 2
- impliziter Länge 54
- Integrität der ~ 184
- Kapselung 5
- nicht initialisierte 133
- numerische 61
- Padding 54
- Standardanmelde~ 245
- terminierte 52
- Übertragung 7
- variabler Länge 52, 62
- Verarbeitung eingehender ~ 102
- voreingestellte Anmelde~ 244
- Datum 55, 61
  - Darstellung 62
- Debugger 123, 264
  - Ansicht 150
  - Fenster 151
- Debugging
  - Symbol 144
  - Technik 289
  - von Anwendungen 264
- Debugging Symbols Package (dSYM) 146
- DEC (Instruktion) 127
- Default Gateway 9, 72

Dekompilierung 125

Denial-of-Service 234

- Lücke 234
  - nicht persistent 234
  - persistent 234

DEP *siehe Data Execution Prevention*

DER *siehe Distinguished Encoding Rules*

DES *siehe Data Encryption Standard*

Destination Network Address Translation (DNAT) 27, 74

- aktivieren 76
- Konfiguration 77

Dezimalzahlen 61

DH *siehe Diffie-Hellman*

DHCP *siehe Dynamic Host Configuration Protocol*

Diffie-Hellman (DH) 182

- Algorithmus 183

Digital Signature Algorithm (DSA) 186

Disassemblierung 125

Disassembly-Fenster 151

Discover 78

Dissector (Sezierer) 15, 106

- entwickeln 109
- für User Datagram Protocol (UDP) 107

Distinguished Encoding Rules (DER) 59

DNAT *siehe Destination Network Address Translation*

DNS *siehe Domain Name System*

DNSMasq 322

dnsspoof 39

- Domain Name System (DNS) 4
  - Server 38–39
- Dotfuscator 160
- dotnet-Binary 88
- Downgrade-Angriff 198
- druckbare Zeichen 48
- DS (Selektorregister) 130
- DSA *siehe Digital Signature Algorithm (DSA)*
- dSYM *siehe Debugging Symbols Package*
- DTLS *siehe Datagram Transport Layer Security*
- DTrace 19
  - Skript 19
- Dynamic Host Configuration Protocol (DHCP) 69, 78
  - Acknowledgment (ACK) 78
  - Discover 78
  - Offer 78
  - Request 78
  - Spoofing 78–79
- dynamische Bibliothek 220
- dynamisches
  - Linking 125–126
  - Reverse Engineering 150

## **E**

- EAX (Mehrzweckregister) 128–129
- EBP (Indexregister) 129
- EBX (Mehrzweckregister) 129
- ECB *siehe Electronic Code Book*
- ECDH *siehe Elliptic Curve Diffie-Hellman*

ECX (Mehrzweckregister) 129  
EDI (Indexregister) 129  
EDX (Mehrzweckregister) 129  
EFAULT 293  
EFLAGS (Kontrollregister) 129  
einfachen Netzwerkclient implementieren 207  
einfachen Server implementieren 209  
einfacher ChatServer 210  
einfacher Paket-Editor 209  
Einmalverschlüsselung *siehe One-Time Pad (OTP)*  
einzelne Pakete betrachten 96  
EIP (Indexregister) 129, 151  
    Fenster 151  
Electronic Code Book (ECB) 172  
    Verschlüsselung 172  
Element 64  
ELF *siehe Executable Linking Format*  
Elliptic Curve Diffie-Hellman (ECDH) 227  
elliptische Kurve 180  
E-Mail 4  
Encoding Layer *siehe Codierungsschicht*  
Endian  
    Big ~ 46  
    Format 46  
    Little ~ 46  
Endianness 46  
Endpunkt-Authentifizierung 195  
Entschlüsselung 226

- Entwicklung von Exploits 320
- Erkennung von Fehlern 2
- errno 293
- ES (Selektorregister) 130
- Escaping 65
- ESI (Indexregister) 129
- ESP (Indexregister) 129
  - Fenster 152
- Ethernet 3
  - Netzwerk 8
- Ettercap 79, 322
- Executable
  - Dateiformat für ~ 132
  - unmamaged 219
- Executable Linking Format (ELF) 133
- exit (Systemaufruf) 291
- Exploit 274
  - Entwicklung 320
  - von Speicherlücke 275
- Exponent
  - öffentlicher 180
  - privater 181
- Extensible Markup Language (XML) 64
- Extensible Messaging and Presence Protocol (XMPP) 65

**F**

- Faktorisierung 180
- Falltürfunktion 180
- falsch *siehe false*

false (falsch) 61

Federal Information Processing Standard (FIPS) 169

Fehler

bei der dynamischen Speicherallokation 244

Code 293

der ASLR-Implementierung 304

Erkennung 2

Korrektur 2

Meldungen mit zu viel Information 249

Off-by-One-~ 240

fehlerhafter Zugriff auf Ressourcen 247

festcodierte Anmeldedaten 244

File Transfer Protocol (FTP) 28

FILETIME (Windows) 55

Financial Information Exchange (FIX) 62

Protokoll 62

finished-Paket 198

FIPS *siehe Federal Information Processing Standard*

FIX *siehe Financial Information Exchange*

Flag

Bestätigungs-~ ACK 46

Bit-~ 46

PROT\_EXEC 287

Fließkommazahlen 45

Flusssteuerung 2

Follow-Stream-Button 94

Follow-TCP-Stream-Ansicht 94, 97

Footer 5

Formate für strukturierten Text 63

Formatierung von Daten 2

Formatstring 255

    Lücke 255

        printf 255

Fragmentierung 57

Frame 7

Framework zum Fuzzing 320

Free List 280

FreeBSD 19

FS (Selektorregister) 130

FTP *siehe File Transfer Protocol*

Funktion

    add() 138

    mit Python aufrufen 224

    mit Strukturparameter 223

    read 135

    String~

        unsichere 239

    write 135

Funktionsmonitor 123

Fuzzing 233, 261

    Framework zum ~ 320

    Test 262

## **G**

Galois Counter Mode (GCM) 174

ganze Zahlen 61

Gateway 69

Default ~ 72  
Standard-~ 72  
Verkehr an ein ~ weiterleiten 78  
GCC *siehe GNU C-Compiler*  
GCM *siehe Galois Counter Mode*  
GDB *siehe GNU Debugger*  
General Public License (GPL) 16  
General-Registers-Fenster 152  
geroutetes Protokoll 69  
GET 34  
GNU C-Compiler (GCC) 221  
GNU Debugger (GDB) 264  
GPL *siehe General Public License* 16  
Graph-Ansicht (graph view) 140  
grundlegende Analyse 93  
GS (Selektorregister) 130  
**H**  
Hash 186  
    Tabelle 253  
    Wert 186  
Hashed Message Authentication Code (HMAC) 189  
    Wert 189  
Häufigkeitsanalyse 165  
Haupt-Benutzerschnittstelle 140  
    ILSpy 155  
Haupt-Thread 134  
HEAD 34  
Header 5

- Heap-Implementierung 280
- Heap-Layout manipulieren 279
- Heap-Überlauf 277
- Hex-Codierung 66
- Hex Dump 97
  - Ansicht, 3 Spalten mit Informationen 96
  - Modus 95
- Hex-Editor 138
- Hex Ray 138
- HMAC *siehe Hashed Message Authentication Code*
- hochprivilegierte Schreiboperation 284
- höchstwertiges Bit *siehe Most Significant Bit (MSB)*
- Hopper 325
- Hops 71
- Host-Header 27, 37
- Host Order (Hostreihenfolge) 47
- Hostreihenfolge *siehe Host Order*
- hosts-Datei 26
- Hping 316
- HTTP *siehe HyperText Transport Protocol*
- HyperText Transport Protocol (HTTP) 4, 27
  - Forwarding-Proxy 35
  - Header 37
  - Proxy 33, 35
    - Einfache Implementierung 35
  - Request 37
  - Reverse-Proxy 37
    - einfache Implementierung 37

Nachteile 40

Vorteile 39

## I

IDA Pro 139, 324

Free Edition 139

IEEE Standard for Floating-Point Arithmetic (IEEE 754) 45

IEEE-Format 45

ILSpy 325

Analyse eines Typs durch ~ 157

Haupt-Benutzerschnittstelle 155

nutzen 155

Suchfenster 156

implizite Länge von Daten 54

Imports-Fenster 146–147

INC (Instruktion) 127

Indexierung

Out-of-Bounds-~ 242

Indexregister

EBP 129

EDI 129

EIP 129

ESI 129

ESP 129

inet\_pton 136

Information, symbolische 144

Inhalte

einer TCP-Session lesen 94

parsen 5

Inhaltsschicht 9

Initialisierungsvektor (IV) 174

- Byte 177
- Original-IV-Byte 178

Instruction Set Architecture (ISA) 127

Instruktion, mnemotechnische 127

Integer-Überlauf 241

Integerwert variabler Länge 44

Integrität 163

- der Daten 184

Intel-Syntax 128

Internet Protocol Suite (IPS) 3

Internetschicht 3

Interpreter 124

interpretierte Sprache 124

IP-Adresse 7

IPS *siehe Internet Protocol Suite*

IPv4 3

IPv6 3

ISA *siehe Instruction Set Architecture*

IV *siehe Initialisierungsvektor*

## **J**

Japanisch 49

JAR *siehe Java-Archiv*

Java 30

- Anwendung 31, 158
- Code wiederverwenden 217
- Bytecode 153

- Klasse laden 219
- Runtime 30
- TCP-Client 31
- Java Decompiler (JD) 159, 323
  - GUI 159
- Java Remote Method Invocation (RMI) 33
- Java-Archiv (JAR) 158
- JavaScript Object Notation (JSON) 64
  - Objekt 64
- Jcc (Instruktion) 127
- JD *siehe Java Decompiler*
- JMP (Instruktion) 127
- JSON *siehe JavaScript Object Notation*
- K**
- Kali Linux 321
- Kanal-Mechanismus 57
- Kanonisierung 247
- Kanonisierungslücke 247
- Kernel-Modus 17
- Key Usage 192
- Key-Scheduling-Algorithmus 169
- Key-Stream 178
- Klartext 164
- Klasse in Java laden 219
- Kollisionsangriff 187
- Kommandozeilen-Utility 36
- Kommunikation zwischen Clients 89
- kompilierte Sprache 125

- Komplexität
  - algorithmische 252
  - Berechnungs~ 253
- Komprimierungsbibliothek ZLib 148
- Konfiguration eines Routers 72
- konfigurierbare Kryptografie 254
- Konstanten, magische 148
- Kontrollregister, EFLAGS 129
- Korrektur von Fehlern 2
- Kryptoanalyse 164
- Kryptografie, konfigurierbare 254
- kryptografischer
  - Algorithmus 149
    - Analyse 150
    - Hash-Algorithmus 184
- Kurve, elliptische 180
- L**
- Längenpräfix 53
- Least Significant Bit (LSB) 42
- Length-Extension-Angriff 187
- LF *siehe Line Feed*
- LibPCAP 313
- Line Feed (LF) 62
- Linking 126
  - dynamisches 125–126
  - statisches 125–126
- Little Endian 46
- LLDB 264

Localhost 14

LSB *siehe Least Significant Bit*

Lua 106

sezieren mit ~ 110

## **M**

MAC *siehe Media Access Control*

MAC *siehe Message Authentication Code*

Mach-O-Format 133

macOS 19

magische Konstanten 148

Mallory 316

Malware 27

Protokoll 11

Managed Code 153

Man-in-the-Middle-Angriff 13, 23, 226

Manipulation lokaler Variablen 306

Maschinensprache 124

Maschine, virtuelle 153

Masquerading *siehe auch Source NAT (SNAT)* 74

Massenspeicherüberlastung 251

Master Secret 197

Maßnahme gegen Speicherlücke 298

MDA *siehe Message Digest Algorithm*

MD5-Algorithmus 148

MD5-Hash-Algorithmus 148, 187

Blockstruktur 188

Media Access Control (MAC) 6

Adresse 6

## Mehrzweckregister

EAX 129

EBX 129

ECX 129

EDX 129

Fenster 152

Memory Exhaustion *siehe Speicherüberlastung*

Message Authentication Code (MAC) 187

Message Digest Algorithm (MDA) 185

## Metasploit

Framework 321

Payload

    Zugriff auf ~ 297

Shell-Code 296

Microsoft Message Analyzer 312

MIME *siehe Multipurpose Internet Mail Extensions*

Minuszeichen (-) 61

MIPS (Prozessorarchitektur) 47

Mitmproxy 319

mnemotechnische Instruktion 127

Modi, alternative 174

Modulo-Arithmetik 241

mono-Binary 88

Mono-Projekt 154

Most Significant Bit (MSB) 42

MOV (Instruktion) 127

Mozilla Firefox 30

MSB *siehe Most Significant Bit*

MS-DOS 132  
Multibyte-Zeichensatz 49  
Multiplexing 57  
Multipurpose Internet Mail Extensions (MIME) 63  
    Nachricht 63  
Multitasking 133  
Mutations-Fuzzer 262

## **N**

Nachrichtenpakete parsen 111  
Namensraum 214  
NASM *siehe Netwide Assembler*  
NAT *siehe Network Address Translation*  
natives C-Äquivalent 222  
.NET-Anwendung, Code wiederverwenden 212  
.NET Reflector 326  
Netcat 202, 317  
Netwide Assembler (NASM) 286  
Network Address Translation (NAT) 74  
    Regeln 77  
Network News Transfer Protocol (NNTP) 65  
Network Order (Netzwerk-Reihenfolge) 47  
Netz des Vertrauens *siehe Web Of Trust (WOT)*  
Netzwerk  
    Adressinformation 58  
    Byteordnung 100  
    Capturing  
        aktives 315  
    einfachen Netzwerkclient implementieren 207

Kommunikation 4–5  
Konnektivität 316  
Protokoll 2, 9, 58  
    Analyse 311  
Proxy 23  
Reihenfolge *siehe Network Order*  
Routing 8  
Schnittstelle 136  
Spoofing 322  
Stack 8  
Substitutions-Permutations-~ 170  
Umleitung 322  
Verkehr generieren 91

Netzzugangsschicht 3  
nicht initialisierte Daten 133  
niedrigstwertiges Bit *siehe Least Significant Bit (LSB)*  
Nmap 317  
NNTP *siehe Network News Transfer Protocol*  
No-Execute (NX) 298  
NULL 295  
NUL-Wert 52  
numerische Daten 61  
Nutzdaten 5  
NX *siehe No-Execute*

**O**

OAEP *siehe Optimal Asymmetric Encryption Padding*  
octet-stream 63  
Off-by-One

- Fehler 240
- Pufferüberlauf 239
- Offenlegung
  - von Benutzernamen 245
  - von Informationen 235
- öffentliche Connect ()-Methode 215
- öffentlicher
  - Exponent 180
  - Schlüssel 179, 181
- Offer 78
- Oktett 42
- One-Time Pad (OTP) 167
- open 21
- Open-Source-Tool Dante 31
- OpenSSL (Verschlüsselungsbibliothek) 147
- Operanden 127
- Optimal Asymmetric Encryption Padding (OAEP) 182
- OR (Instruktion) 127
- Oracle-Angriff 176
- OSI-Modell 15
- OTP *siehe One-Time Pad*
- Out-of-Bounds-Indexierung 242
- P**
- package-private 217
  - Klasse 217
- Packet 6
- Padding
  - Block

- äußerer 189
- innerer 189
- Oracle Attack 176
- von Daten 54
- Wert 177
- Page Heap 273
- Paket
  - change cipher spec-~ 198
  - Editor, einfacher 209
  - einzelne Pakete betrachten 96
  - erfassen 91
  - finished-~ 198
  - Generierung 320
  - Nachrichtenpaket parsen 111
  - Prüfsumme eines Pakets 104
  - Sniffer 14
  - Sniffing 16
  - Struktur identifizieren 95
  - Transmission Control Protocol (TCP)-~ 96
- Parsen
  - der Inhalte 5
  - eines Nachrichtenpakets 111
  - von Protokollen 119
- Parser-Code für Proxy 118
- parser.csx-Skript 206
- partielles Überschreiben 304
- passives Capturing 13, 22, 311
  - Techniken 16

Payload ausführen 298

P-Box *siehe Permutationsbox*

PDB-Datei (program database) 144

PDU *siehe Protocol Data Unit*

PE *siehe Portable Executable*

PEiD 149

PEM-Format 227

Perfect Forward Secrecy 200

Permutationsbox (P-Box) 171

PGP *siehe Pretty Good Privacy*

PKCS#7 *siehe Public Key Cryptography Standard #7*

PKI *siehe Public-Key-Infrastruktur*

plain 63

Pluszeichen (+) 61

Point-to-Point-Protokoll (PPP) 3

POP (Instruktion) 127

Port 4

- Forwarding 24
- Proxy 24, 27
- Nummer 6

Portable Executable (PE) 132

POSIX 17

- Zeit 55

POST 34

PowerPC 42

PPP *siehe Point-to-Point-Protokoll*

Pre-Master Secret 197

Pretty Good Privacy (PGP) 190

- printf-Formatstring 255
- private Connect()-Methode 215
- privater
  - Exponent 181
  - Schlüssel 179, 181
- Process Monitor 21
- Programmfluss 131
- Programmiersprache
  - speichersichere 236
  - speicherunsichere 236
- ProGuard 160
- Promiscuous Mode 14
- Protocol Data Unit (PDU) 5
- Protokoll 62
  - Analyse mittels Proxy 117
  - geroutetes 69
  - Malware-~ 11
  - mit Python sezieren 100
  - Modell 10
  - parsen 119
  - Remote Procedure Calls (RPC)-~ 26
  - Stack 3
  - Struktur ermitteln 97
  - Test 316
  - textbasiertes 60
  - Verhalten ändern 120
- Proxifier 31
- Proxy 23

- HTTP-~ 33, 35
  - Einfache Implementierung 35
- HTTP-Forwarding-~ 35
- HTTP-Reverse-~ 37
  - einfache Implementierung 37
  - Nachteile 40
  - Vorteile 39
- Konfiguration in Firefox 30
- Netzwerk-~ 23
- Parser-Code für ~ 118
- Port-Forwarding-~ 24, 27
- Protokollanalyse 117
- SOCKS-~ 28, 30
  - einfache Implementierung 29
- Verkehr auf ~ umleiten 25, 38
- Verkehrsanalyse 114
- Prozentcodierung 66
- Prozess 133
- Prozessorarchitektur
  - ARM 47
  - MIPS 47
  - SPARC 47
- Prüfsumme 185
  - berechnen 103
  - eines Pakets 104
- Pseudoregister \$pc 267
- Pseudozufallszahlengenerator 167
- Public Key Cryptography Standard #7 (PKCS#7) 175

- Padding 175
- Public-Key-Information 191
- Public-Key-Infrastruktur (PKI) 190
- Public-Key-Verschlüsselung 179
- Pufferüberlauf 237
  - bei fester Pufferlänge 237
  - bei variabler Pufferlänge 237, 240
  - Off-by-One-~ 239
- PUSH (Instruktion) 127
- Python 99, 236
  - Bibliothek laden 221
  - ctypes 222
  - Funktion aufrufen 224
  - komplizierte Funktion aufrufen 222
  - Protokoll mit ~ sezieren 100

## **Q**

- Quelladresse 6, 22
- Quellcode 124

## **R**

- rand() (C-Bibliotheksfunktion) 168
- RAX-Register 288
- RDP *siehe Remote Desktop Protocol*
- RE *siehe Reverse Engineering*
- read 18, 21, 135
- Real Time Messaging Protocol (RTMP) 33
- recv 18, 135
- recvfrom 18
- Reflector 155

- Reflexion 211
- Reflexions-API nutzen 213
- Reflexionstypen
  - in Java 218
  - unter .NET 213
- Register
  - CPU-~ 128
  - Index~ 129
    - Fenster 151–152
  - Kontroll~ 129
  - Mehrzweck~ 129
    - Fenster 152
  - RAX-~ 288
  - Selektor~ 130
- Reihenfolge
  - Byte~ (Endianness) 46
  - Host~ (Host Order) 47
  - Netzwerk-~ (Network Order) 47
- Remote Code Execution 234
- Remote Desktop Protocol (RDP) 57
- Remote Method Invocation (RMI) 33
- Remote Procedure Calls (RPC) 26
  - Protokoll 26
- Replay 201
  - von UDP-Verkehr 204
- Request 78
- Rerouting von Verkehr 69
- RET (Instruktion) 127

- RETN (Instruktion) 127
- Return-Oriented Programming (ROP) 300
  - Gadget 301
- Ret2Libc 300
- Reverse Engineering (RE) 123, 323
  - dynamisches 150
  - Ressourcen 161
    - ELF-Dateiformat 161
    - macOS Mach-O-Format 161
    - OpenRCE-Foren 161
    - PE-Dateiformat 161
  - statisches 138
- Reverse-Shell 297
- Rich Site Summary (RSS) 65
- RMI *siehe Remote Method Invocation*
- Root-Zertifikat 191
  - vertrauenswürdiges 228
- ROP *siehe Return-Oriented Programming*
- Router 69
  - Konfiguration 72
- Routing
  - aktivieren
    - Windows 73
    - \*nix 73
  - Tabelle 9, 71
- RPC *siehe Remote Procedure Calls*
- RSA
  - Algorithmus 180

Padding 182

Schlüssel 167

Signatur 186

RSS *siehe Rich Site Summary*

RTMP *siehe Real Time Messaging Protocol*

Ruby 236

Runtime 153–154

## **S**

S-Box *siehe Substitutionsbox*

Scapy 321

Schlüssel 164

    Austausch nach Diffie-Hellman 182

    öffentlicher 179, 181

    privater 179, 181

    Session~ 182

Schreiboperation

    hochprivilegierte 284

    mit niedrigen Rechten 285

    willkürliche 283

Secure Hashing Algorithm (SHA) 185

Secure Sockets Layer (SSL) 193

Segment 6, 96

selbst signiertes Zertifikat 191

Selektorregister

    CS 130

    DS 130

    ES 130

    FS 130

- GS 130
- SS 130
- send 18, 135
- sendfrom 18
- Serpent 171
- Server
  - einfachen Server implementieren 209
  - starten 88
  - Zertifikat 195
- Sessionschlüssel 182
- Sessionzustand 2
- sezieren mit Lua 110
- Sezierer *siehe Dissector*
- SGML *siehe Standard Generalized Markup Language*
- SHA *siehe Secure Hashing Algorithm*
- SHA-1-Hash-Algorithmus 187
- Shell-Code 286, 290
  - Metasploit 296
- shell\_bind\_tcp 297
- shell\_reverse\_tcp 297
- SHL (Instruktion) 127
- SHR (Instruktion) 127
- Sicherheit 163
- Sicherheitsanforderung 198
- Sicherheitslücke 261
  - ausnutzen 274
  - Offenlegung von Speicherinformationen 302
- Sicherheitsproblem aufspüren 261

- Sicherheitsprodukte 27
- Signatur 164, 184
  - Algorithmus 184
    - asymmetrischer 186
  - RSA~ 186
- Signed Integer 43
- Simple Mail Transport Protocol (SMTP) 4
- Simple Network Management Protocol (SNMP) 59
- Skriptsprache 124
- Slash (/) 89
- SMTP *siehe Simple Mail Transport Protocol*
- SNAT *siehe Source NAT*
- SNMP *siehe Simple Network Management Protocol*
- Socket 136
- socket 18
- SOCKS-Proxy 28, 30
  - einfache Implementierung 29
- SOH *siehe Start of Header*
- Solaris 19
- Source NAT (SNAT) 74
  - aktivieren 74
  - unter Linux konfigurieren 75
- SPARC (Prozessorarchitektur) 47
- Speicher
  - Abschnitt 133
  - Allozierung in definierten Speicherpools 282
  - Allozierung, dynamische
    - Fehler 244

- Lücke
  - Maßnahme gegen Speicherlücke 298
  - zur Informationsgewinnung 303
- Massenspeicherüberlastung 251
- Sicherheitslücke durch Offenlegung von Speicherinformationen 302
- Überlastung (memory exhaustion) 250
- Überlastungsangriff 250
- Verfälschung 236
- verschwendeter 280
- speichersichere Programmiersprache 236
- speicherunsichere Programmiersprache 236
- Sprache
  - Assembler~ 125
  - interpretierte 124
  - kompilierte 125
  - Maschinen~ 124
  - Programmier~
    - speichersichere 236
    - speicherunsichere 236
  - Skript~ 124
  - Structured Query Language (SQL) 257
- SQL *siehe Structured Query Language*
- SS (Selektorregister) 130
- SSL *siehe Secure Sockets Layer*
- Stacktrace 268
- Stacküberlauf 275
  - erkennen 305
- Stackunterlauf 308

- Canaries umgehen 307
- Stackvariable 143
  - analysieren 143
- Standard Generalized Markup Language (SGML) 64
- Standardanmeldedaten 245
- Standard-Gateway 72
- Start of Header (SOH) 62
- statisches
  - Linking 125–126
  - Reverse Engineering (RE) 138
- stdcall 224
- Steuerzeichen 48
- strace 18
  - Utility 18
- String
  - analysieren 147–148
  - Funktion, unsichere 239
- Strip-Tool 146
- Stromchiffre 168, 178
- struct-Bibliothek 100
- Structured Query Language (SQL) 257
  - Injektion 257
- strukturierter Text, Formate 63
- Strukturparameter 223
- SUB (Instruktion) 127
- Subroutine 131
- Subroutinenaufruf 131
- Substitutionsbox (S-Box) 171

Substitutionschiffre 165  
Substitutions-Permutations-Netzwerk 170  
Sulley 322  
SuperFunkyChat 87  
Switch 7  
symbolische Information 144  
symmetrische Verschlüsselung 168

## System

Aufruf 17–18, 290

  exit 291

  recv 135

  send 135

  Tracing von Systemaufrufen 17

  write 292

  unixoides 17

system() 300

## T

Tag-Wert ermitteln 104

Tag, Länge, Wert *siehe Tag, Length, Value (TLV)*

Tag, Length, Value (TLV) 56

  Protokoll 59

  Wert 56

TCP *siehe Transmission Control Protocol*

TCPDump 313

TcpNetworkListener 156

TCP/IP 3

TDES *siehe Triple DES*

terminierte Daten 52

- terminierter Text 62
- TEST (Instruktion) 127
- Testdatensatz generieren 263
- Text
  - Codierung, Zeichenersetzung 258
  - Formate für struktuierten ~ 63
  - mit Trennzeichen 62
  - terminierter 62
- textbasiertes Protokoll 60
- Thread 133
  - Haupt~ 134
- TLS *siehe Transport Layer Security*
- TLV *siehe Tag, Length, Value*
- Token 62
- traceconnect.d 20
- Traceroute 70
- traceroute 70
- tracert 70
- Tracing von Systemaufrufen 17
- Transmission Control Protocol (TCP) 3
  - Clientverbindung mit einem TCP-Server herstellen 136
  - Follow-TCP-Stream-Ansicht 94, 97
  - Inhalte einer TCP-Session lesen 94
  - Paket 96
  - Stream-Parser 114
- Transport Layer Security (TLS) 193, 224
  - Handshake 194
    - Prozess 194

Record-Protokoll 194  
TLS 1.2 erzwingen 227  
Verkehr entschlüsseln 226  
Verschlüsselung 224  
Transport Layer *siehe Transportschicht*  
Transportschicht 3–4, 10  
Trennzeichen 62  
Triple DES (TDES oder 3DES) 170  
true (wahr) 61  
Tshark 202  
Twofish 171

**U**

Überschreiben, partielles 304  
UCS *siehe Universal Character Set*  
UDP *siehe User Datagram Protocol*  
Uhrzeit 55, 61  
UI *siehe User Interface*  
unbedingte Verzweigung 131  
Unicode 49  
    Codierung  
        UTF-16 50  
        UTF-32 50  
        UTF-8 50  
    Zeichensatz 258  
Unicode Transformation Format (UTF) 49  
Unified-Sniffing-Modus 83  
Uniform Request Identifier (URI) 34  
Universal Character Set (UCS) 49

Unix 55

    Zeit 55

unixoides System 17

unmanaged Executable 219

unsichere Stringfunktion 239

Unsigned Integer 42

URI *siehe Uniform Request Identifier*

Use-after-free-Lücke 278

User Datagram Protocol (UDP) 4

    Client zum Senden von Netzwerk-Captures 205

    Dissector 107

    Replay von UDP-Verkehr 204

User Interface (UI) 5

User-Modus 17

UTF *siehe Unicode Transformation Format*

## **V**

variable Länge von Daten 52, 62

Verarbeitung eingehender Daten 102

Verfälschung des Speichers 236

Verisign 191

Verkehr

    an ein Gateway weiterleiten 78

    ausgehender 98

    Capturing von Beispiel~ 206

    Netzwerkverkehr generieren 91

    Rerouting von ~ 69

    TLS-Verkehr entschlüsseln 226

    umleiten 25, 38

- Verkehrsanalyse 114
- Verschleierungstaktik 160
- Verschlüsselung 164
  - asymmetrische 179
  - ECB-~ 172
  - Einmal-~ *siehe One-Time Pad (OTP)*
  - mit TLS 224
  - Public-Key-~ 179
  - symmetrische 168
  - XOR-~ 120–121, 166
- Verschlüsselungsalgorithmus 164
- Verschlüsselungsbibliothek
  - Crypt32.dll 147
  - OpenSSL 147
- verschwendeter Speicher 280
- Vertrauenskette 197
- vertrauenswürdige Root-Zertifikat 228
- Vertraulichkeit 163
- Verzweigung
  - bedingte 131
  - unbedingte 131
- Virtual Function Table (VTable) 271
- VirtualAlloc 279
- virtuelle Maschine 153
- voreingestellte Anmeldedaten 244
- vorzeichenbehaftete ganze Zahl *siehe Signed Integer*
- vorzeichenlose Zahl *siehe Unsigned Integer*
- VTable *siehe Virtual Function Table*

Vulnerabilitätsklasse 234

## **W**

Wagenrücklauf *siehe Carriage Return (CR)*

wahr *siehe true*

Web Of Trust (WOT) 190

Modell 190

Webanwendung testen 318

wget 36

Whitespace 62

willkürliche Schreiboperation 283

Windows

FILETIME 55

Routing aktivieren 73

XP SP2 302

Winsock-Bibliothek 134

Wireshark 14, 90, 105–106, 314

Capturing mit ~ 93

Conversations-Fenster 93

Follow-TCP-Stream-Ansicht 94, 97

grundlegende Analyse 93

Hauptfenster 90

Standardansicht 15

TCP-Stream-Parser 114

WOT *siehe Web Of Trust*

write 18, 21, 135

Systemaufruf 292

## **X**

XML *siehe Extensible Markup Language*

XMPP *siehe Extensible Messaging and Presence Protocol*

XOR

Operation 166

Parameter 120

Verschlüsselung 120–121, 166

XOR (Instruktion) 127

XP SP2 302

XSS *siehe Cross-Site Scripting*

XXD 99

xxd (Tool) 204

x86 47

Architektur 126

Instruktion 127

Mnemonic 127

ADD 127

AND 127

CALL 127

CMP 127

DEC 127

INC 127

Jcc 127

JMP 127

MOV 127

OR 127

POP 127

PUSH 127

RET 127  
RETN 127  
SHL 127  
SHR 127  
SUB 127  
TEST 127  
XOR 127

X.500-Namen 196

X.509-Zertifikat 59–60, 190

## **Z**

Zahlen

Dezimal~ 61

ganze ~ 61

ZAP *siehe Zed Attack Proxy*

Zed Attack Proxy (ZAP) 319

Zeichencodierung 49

Zeichenersetzung bei Textcodierung 258

Zeichen-Mapping 49

Zeichensatz

Multibyte~ 49

Unicode~ 258

Zeilenendezeichen 62

Zeilenvorschub *siehe Line Feed (LF)*

Zertifikat

Certificate Authority (CA) 191

Client~ 197

ersetzen 227

Root~ 191

- vertrauenswürdiges 228
- selbst signiertes 191
- Server~ 195
- X.509~ 59–60, 190
- Zertifikatskette 192
- Zertifikatsmanager 229
- Zertifikats-Pinning 199
- Zertifikatsspeicher 230
- Zertifikatssperrliste *siehe Certificate Revocation List (CRL)*
- Zertifikats-Subject 196
- Zieladresse 6, 22
- ZLIB (Komprimierungsbibliothek) 148
- Zufallszahlengenerator 167
- Zugriff auf Ressourcen, fehlerhafter 247
- Zweierkomplement 43

## **Ziffern**

- 3DES *siehe Triple DES*

### 32-Bit

- System, relative Adresse 294

- Wert 45

- Wort 44

### 64-Bit

- System, relative Adresse 294

- Wert 45

### 7-Bit-Integer 44

### 8086

- CPU 126

- Prozessor 126

## **Sonderzeichen**

/ Slash 89

\ Backslash 52

- Minuszeichen 61

+ Pluszeichen 61

\$pc (Pseudoregister) 267