SS 2012

Fachpraktikum 01599

IT-Sicherheit
Gruppe Wilddiebe 3

Autoren

Stefan Götz, Jürgen Klemm, Thorsten Klingert,
Thomas Koch, Gerald Luber, 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

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Einleitung und Aufgabenstellung

2 OpenVPN

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 Orga-

nisation 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 Zielcomputers 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 Ha-

cker 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, 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

nmap

3.3 Dateiserver hacken

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

3.3.1 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:

```
#!/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

```
#!/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

```
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.

3.3.2 Zugriff auf den Dateiserver

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

```
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: Script für dynamischen Portscan: nmap.sh

172.16.14.40 versucht, sich die Freigaben anzuzeigen. Es war zu diesem Zeitpunkt nur public zu sehen. Auf das public Verzeichnis kann von einem Linux oder Free-BSD 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’
```

3.3.3 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(? @TODO) eingesetzt werden kann. Das bekannteste Teilprojekt ist das Metasploit Framework, ein Werkzeug zur Entwicklung und Ausführung von Exploits gegen verteilte Zielrechner. Andere wichtige Teilprojekte sind das Shellcode-Archiv und Forschung im Bereich der IT-Sicherheit.

3.4 Webserver hacken

Wir wissen bereits, dass der Webserver von *Mayer Brot* unter der Adresse 172.16.14.30 zu erreichen ist[1]. 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 nicht überprüfen, und gibt eine entsprechende Warnung aus. Wenn wir diese ignorieren, gelangen wir zur Homepage von *Mayer Brot* (siehe Abbildung 3.4).

[1] vgl. den Portscan aus Abschnitt 3.2

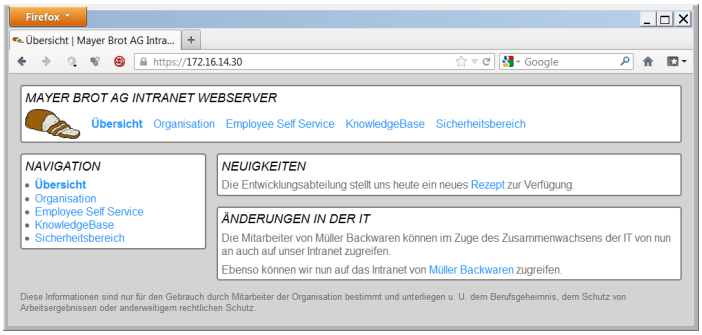


Abbildung 3.4: Homepage von *Mayer Brot*

Das erste Rezept ist nicht weiter geschützt. Über den entsprechenden Link unter NEUIGKEITEN kann das verschlüsselte Rezept heruntergeladen werden[2].

Auf der Homepage befindet sich auch ein *Sicherheitsbereich*. Ein Zugriff darauf wird mit einer Fehlermeldung quittiert (siehe Abbildung 3.5). Hier ist wohl das zweite Rezept zu finden. Wenn wir ins Blaue hinein raten und `https://172.16.14.30/private/rezept.txt` eintippen, kommen wir auch nicht weiter.

[2]Die Entschlüsselung wird in Abschnitt 4.4 gezeigt

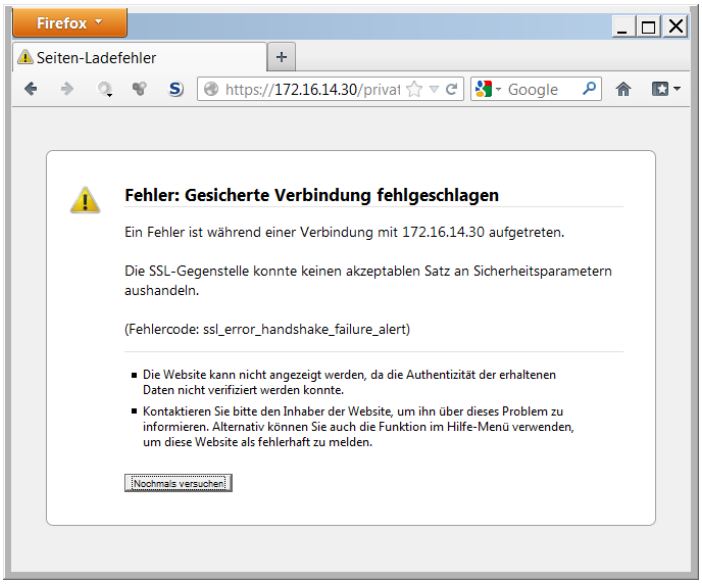


Abbildung 3.5: Gesicherte Verbindung fehlgeschlagen

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 (siehe Abschnitt 3.2) haben wir schon eine Liste der offenen Ports erhalten (siehe Tabelle 3.1).

Tabelle 3.1: Offene Ports des Webservers

Port	name	proto	info
22		tcp	
80	http	tcp	Apache (302-https://172.16.14.30/)
443	https	tcp	
2049	nfsd	udp	NFS Daemon 100005 v1

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 su-

chen.

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 3.6). Wenden wir diesen Exploit an, führt das aber irgendwie nicht zum gewünschten Erfolg.

Abbildung 3.6: Webserver Exploit

In *Metasploit* wird jedem Exploit ein Rang zugeordnet. Damit wird der Wirkungsgrad und 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 unser 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.

3.4.2.1 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 Version gesucht (vgl. [Nik]).

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

Tabelle 3.2: Nikto Warnungen

OSVDB[3]	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.

3.4.2.2 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 *Au-*

[3] Abkürzung für *Open Source Vulnerability Database* (siehe `http://www.osvdb.org`)

dit und *Discovery* (siehe Abbildung 3.7). Mit *Discovery* veranlassen wir die Schwachstellen-Analyse und mit *Audit* suchen wir entsprechende Exploits. Danach starten wir den Scan-Vorgang.

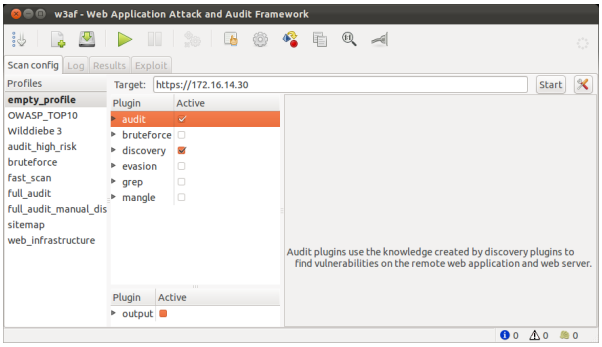


Abbildung 3.7: W3AF Scan config

Für den Analyse wird ein Proxy mit dem Namen *spiderMan* gestartet, der über `127.0.0.1:44444` zu erreichen wird. Wir konfigurieren *Firefox* so, dass die Kommunikation über diesen Proxy abläuft. Anschließend können wir die Homepage von *Mayer Brot* aufgerufen 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.

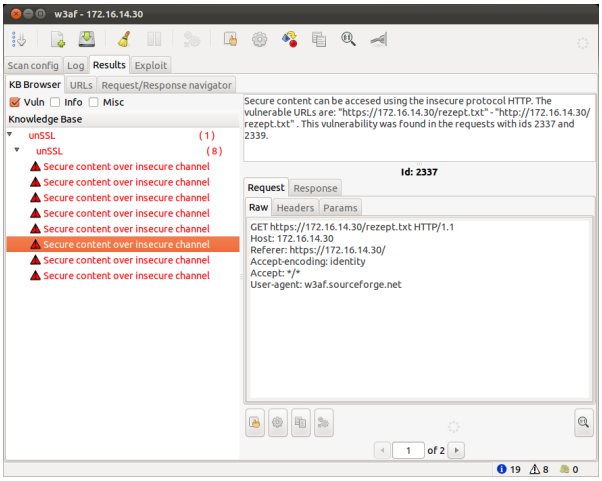


Abbildung 3.8: W3AF Results

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. Den 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[4]. Maximale Kontrolle über den Dateiserver erhalten werden, 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 3.9).

[4] dazu gehen wir wie in Abschnitt 3.3 vor

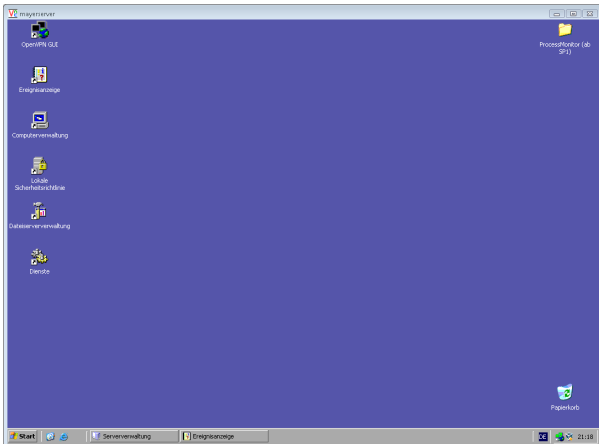


Abbildung 3.9: VNC-Sitzung mit Dateiserver

Auf dem Desktop ist das Icon von *OpenVPN GUI* zu sehen. Aus unseren Erfahrungen mit *OpenVPN GUI*[5] wissen wir, dass im Konfigurationsverzeichnis auch das Zertifikat des Dateiservers zu finden. Es kann nicht schaden, das komplette Konfigurationsverzeichnis auf unseren Rechner zu kopieren.

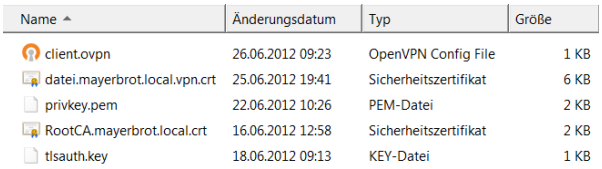
Anschließend kümmern wir uns um das geheime Rezept. Dazu versuchen wir die Homepage des Webservers über den Internet-Explorer zu erreichen. Leider erscheint eine Fehlermeldung, dass die Datei nicht gefunden werden kann. Die Seite von Google kann aber aufgerufen werden. Ein Ping auf den Webserver klappt 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 wird vom Dateiservers die kompletten OpenVPN-Konfigurationsdateien heruntergeladen (vgl. Abbildung 3.10). Darunter befin-

[5] vgl. Abschnitt 2

det sich auch das Zertifikat des Dateiservers. Bei einem Zertifikat handelt sich um einen digitalen Ausweis. Verwenden wird 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 prüfen, ob wir damit auf den geschützten Bereich des Webservers 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 3.10: OpenVPN-Konfigurationsdateien des Dateiservers

Um ein Zertifikat in *Firefox* zu importieren muss es als PFX-Datei vorliegen. 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 wird zur Registerkarte „Ihre Zertifikate“ und importieren die Datei `mayerdatei.pfx`. Als Ergebnis erhalten wir das Fenster in Abbildung 3.11.

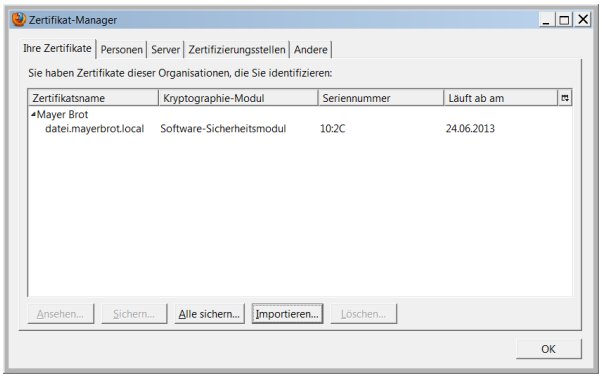


Abbildung 3.11: Importiertes Zertifikat

Wenn wir jetzt auf die Homepage des Webservers zugreifen wird zunächst nach dem Zertifikat gefragt (vgl. Abbildung 3.12). Hier wählen wir das Zertifikat des Dateiservers.

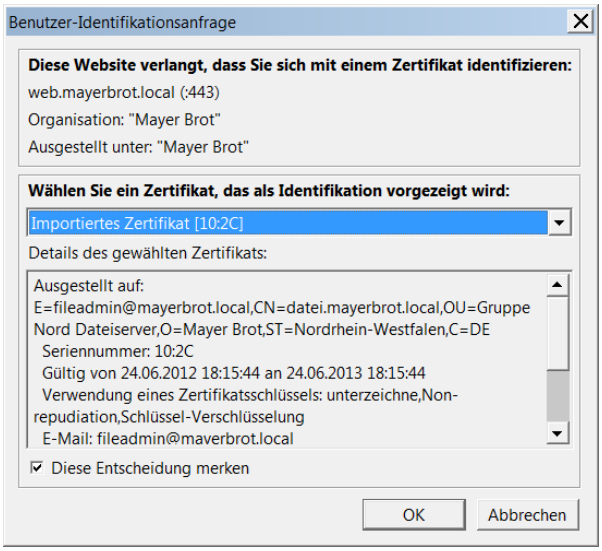


Abbildung 3.12: Zertifizierungsanfrage

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 wird das verschlüsselte Rezept aus dem Sicherheitsbereich[6] (vgl. Abbildung 3.13). Damit sehen wir, dass es durchaus reicht, ein Rechner zu hacken, um Zugriff auf weitere Rechner im Netz zu erhalten.

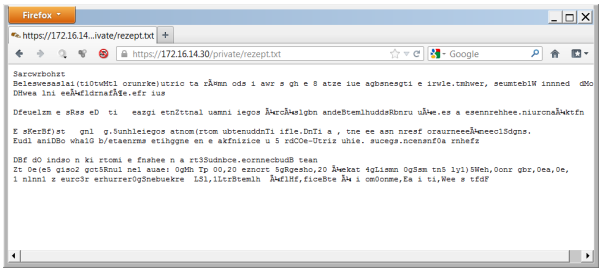


Abbildung 3.13: Zweites Rezept des Webservers

[6] Die Entschlüsselung des Rezepts wird in Abschnitt 4.5 gezeigt

Abbildung 4.1: Der verschlüsselte Text von Rezept 1

4 Entschlüsseln

4.1 Grundlagen

4.2 Rezept 1 entschlüsseln

Um einen chiffrierten Text zu entschlüsseln beginnt man grundlegend mit der Kryptoanalyse.

4.2.1 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.

4.2.1.1 Häufigkeitsanalyse

In jeder Sprache kommen die einzelnen Buchstaben in einem ausreichend langen, natürlichen Text in einer für die Sprache charakteristischen Häufigkeit vor (siehe Abbildung 4.2). Im Deutschen ist der am häufigsten vorkommende Buchstabe das E mit einer Häufigkeit von etwa 17%, gefolgt vom N mit etwa 10%.

“ “	“ “	“ “	,	-	.	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	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

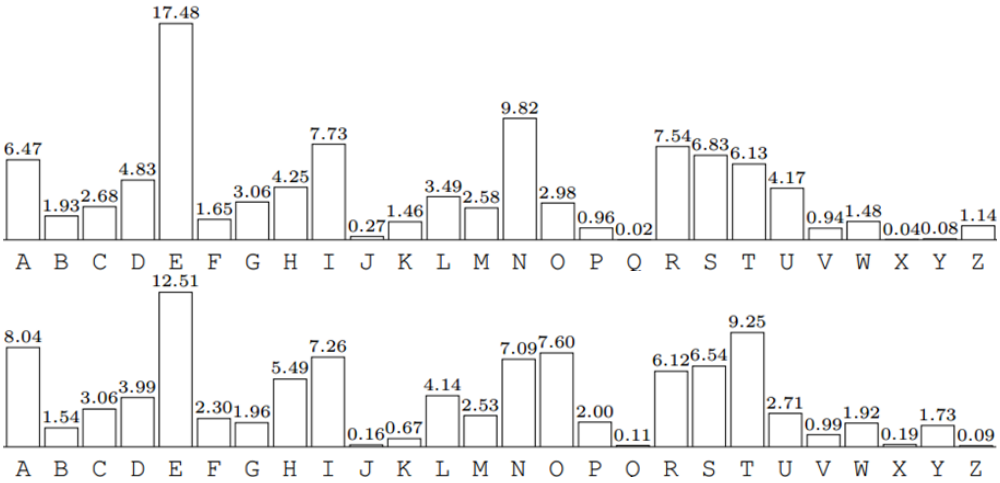


Abbildung 4.2: Häufigkeitsverteilung von Buchstaben im Deutschen (oben) und Englischen (unten) [Kö, kap2.pdf, s. 30]

einem Onlinetool[7] gezählt und ist in Tabelle 4.2.1.1 wiedergegeben.

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

4.2.1.2 Cesar Chiffre

Der Name Caesar-Chiffre stammt von dem gleichnamigen Feldherren Gaius Julius Cesar (100 v. Chr. – 44 v. Chr.).1 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:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
DEFGHIJKLMNOPQRSTUVWXYZABC
```

[7] http://www.woerter-zaehlen.de/index.php

```
ZNAQRYFGHGRA  -> WKXNOVCDEDOX
```

4.2.1.3 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
```

Listing 4.1: Quellcode rot13.c

```
/*
 * Rot-13 encoder/decoder in C
 **/

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

int rot13( int c );

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

  return 0;
}
```

Mandelstuten Mehl, Zucker, 2 Eier, Bittermandelaroma, gehackte 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 lauwarme Milch, 60g Zucker, 1 Würfel Hefe, 4 Eier, 50g weiche Butter, 3 Tropfen Bittermandelöl, 150g gehackte Mandeln, 4 EL lauwarme Milch

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

4.3 Rezept 2 entschlüsseln

Listing 4.2: Das verschlüsselte 2. Rezept: Rezept2NR.txt

```
HrSbkesnuz1eWbsgteeamhWbezldsiDeSgeraeuWgmneSnmmunnant
solsrivtieuMsusrnk3eieteean0vzBribAFhkau2gh15reailsron
n0eeeinceraa1zhsreboeonlnednnaudBciet0himareesbekqlnsn
WleicemareknzinanasegenisznaneDnetdsgnlnsarurieadc3teb
imeptreleDateSreaVkrhuaSfaMnneeeenftenteugabatomeacedS
dnZmtmauhnsnBcea2GCoiesraiMnnanuroembhlsZtn5ogele17get
WisuegmkeWe1Vlrwem5gnnmnn2ene2uhzngohcWle0l1leeWeietin
euereLimndnheni0lseserieadcullsDenstonesedckreesesarbs
eesueaebnSuegRgmhdaseteshbek1udeZeteaghasiSedaeidolnen
dazctuernnneettdihaettKefgbbenntnbimrprrelsDnkfu2rdren
aodn0ueceserneалnsnae0RemTp51Suegznet30atas7gloezelSnb
```

```
 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 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
```

```
ukr5Lsm5Bwi5gbekensga0misasdDnmneeewmmssgneeaiskdseuhs
WasNdteaioedsvcgtnirresenuglmdlhtkIgemlseneduueeegaeof
ahdt5tkdmneetgny0aier0ls5kiholegage0gtsS0ss
```

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

Die Häufigkeitsanalyse ergab, dass ë"wieder am häufigsten auftritt. Das dechiffrieren des Textbeginns mit der Casesar Chiffre ergibt ein sinnloses “EOPYHBPKRW1”, rot-13 ergibt “UEFOXRFAHM1”.

4.3.1 Rail Fence Cipher

iese 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
```

Listing 4.3: rafeci.c

```
#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 i = 0;
int i1 = 0;
int i2 = 0;

int main()
{
  int c;

  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␣␣␣␣␣", alphabet125[i2]);
}

printf("\n");
printf("␣");

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

printf("\n");
printf("␣␣");

for ( i2 = 0; i2 < LIM2; i2++ ) {
  printf("%c␣", alphabet248_2[i2]);
  printf("%c␣␣␣", alphabet248_2[++i2]);
}

printf("\n");
printf("␣␣␣");
```

```
  for ( i2 = 0; i2 < LIM3; i2++ ) {
    printf("%c␣␣␣␣␣", alphabet124[i2]);
  }

  return 0;
}
```

Das entschlüsselte Rezept lautet:

Heidebrot

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 Salzfracht 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 zunächste einer Häufigkeitsanalyse in JCrypTool[8] unterzogen. Als Einstellungen wurden “Monoalphabetisch” sowie als Referenzverteilung “DEUTSCH Descartes” ausgewählt. Als ursprüngliches Alphabet des Klartextes wurde “Printable ASCII” genommen. Die Häufigkeitsanalyse zeigt Abbildung 4.3.

[8] Java Lehrsoftware für Kryptologie, erhältlich unter http://www.cryptool.org

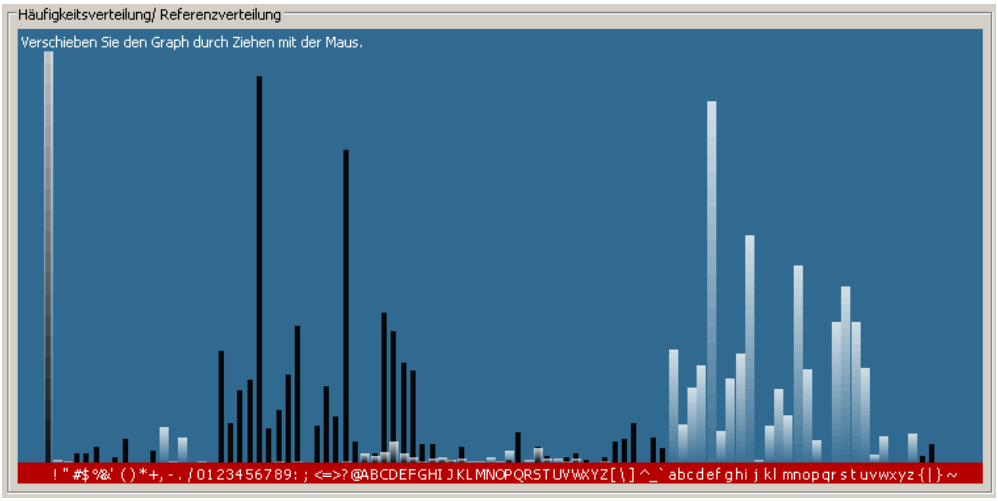


Abbildung 4.3:

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 4.4).

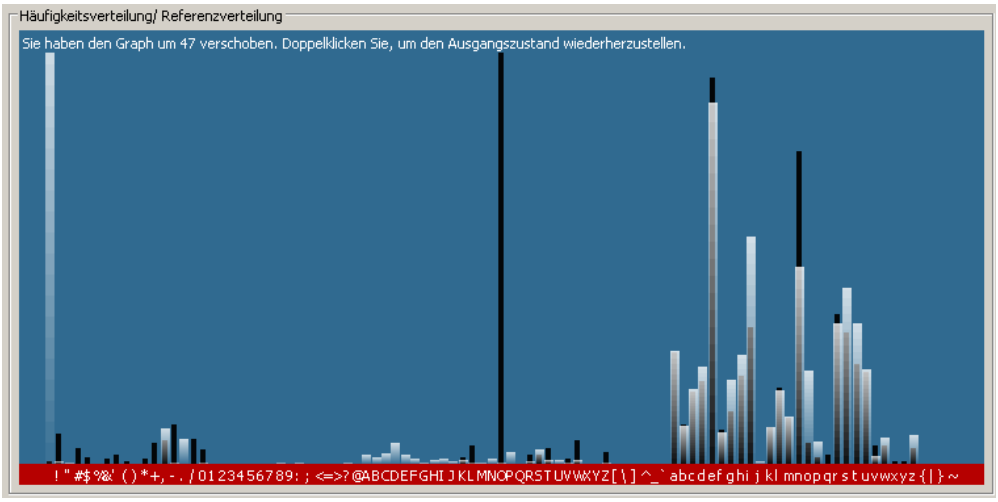


Abbildung 4.4:

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 (Abschnitt 4.4).

```
!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

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 Cesar-Chiffre mit einer Verschiebung um 47 Zeichen.

4.5 Rezept 4 entschlüsseln

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

[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