



TECHNISCHE  
UNIVERSITÄT  
WIEN  
VIENNA  
UNIVERSITY OF  
TECHNOLOGY

# Security VU

183.124

WS2003/2004

## Lab 1 Arbeitsprotokoll

Haider Gerald (0125638)  
Radl Christoph (0102799)  
Schachinger Josef (0125692)

# Inhaltsverzeichnis

<b>AUFBAU DER ARBEIT</b>	<b>3</b>
<b>EINFÜHRUNG</b>	<b>3</b>
<b>1 UNTERSUCHEN DEN NETZWERKS - NETZWERKSTRUKTUR</b>	<b>4</b>
1.1 Feststellung der im Netz aktiven Rechner	4
1.2 Feststellung geöffneter Ports und des Betriebssystems:	5
1.3 Ermitteln der Netzwerktopologie	7
<b>2 FIREWALK-SCANS</b>	<b>8</b>
2.1 Scan gegen 172.16.0.2	8
2.2 Scan gegen 172.16.0.3	9
<b>ZUSAMMENFASSUNG</b>	<b>10</b>
<b>REFERENZEN</b>	<b>10</b>

## Aufbau der Arbeit

Dieses Arbeitsprotokoll beschränkt sich auf Schlussfolgerungen und Erkenntnisse aus den einzelnen Scans. Die detaillierten Ergebnisse können in zwei weiteren Dokumenten eingesehen werden.

Anhang\_Output → nmap und firewalk

nessus\_security.txt → nessus

Die traceroute Ergebnisse werden nur spartanisch in diesem Dokument angeführt, die Erkenntnis daraus bilden sich im Netzwerkdiagramm ab.

## Einführung

Vorbedingungen für unsere Arbeit „Lab1“:

Wir haben mittels SSH (Secure Shell) Zugang zu dem Rechner security.rise.tuwien.ac.at. Für unsere Arbeit haben wir dabei den frei verfügbaren SSH Client PuTTY verwendet. Unsere Aufgabe besteht bei diesem Teil der Übung darin, die Scan-Phase des „Einbrechens“ in Computersysteme zu absolvieren. Dabei ist es für uns zunächst wichtig zu wissen welche Systeme an unseren Ausgangsrechner angeschlossen sind und in weiterer Folge wie diese untereinander verbunden sind. Außerdem wollen wir auch genau herausfinden welche Dienste auf diesen Rechnern laufen.

Wir gehen also gemäß den in [1] eingeteilten Phasens des Hackens davon aus, dass wir die sog. Reconnaissance-Phase bereits absolviert haben. Benutzername und Passwort für das Ausgangssystem haben wir uns also durch intensives „Social Engineering“, „Dumpster Driving“ (oder Umgangssprachlich: „Miststirln“) oder ähnliches verschafft.

# 1 Untersuchen den Netzwerks - Netzwerkstruktur

Zuerst mussten wir herausfinden mit welchen Rechnern wir es in diesem Netzwerk zu tun haben vorgegeben wurden uns dabei nur die im Netzwerk verfügbaren Teilnetze:

```
10.0.0.x
172.16.0.x
192.168.0.x
192.168.1.x
192.168.2.x
192.168.3.x
192.168.4.x
192.168.5.x
```

## 1.1 Feststellung der im Netz aktiven Rechner

Wir stellen im Folgenden, fest welche Rechner von unserem Startsystem (security.rise.tuwien.ac.at) aus direkt oder auf Umwegen erreichbar sind: Dazu bedienen wir uns des beliebten Port Scanners nmap[2]. Allerdings nutzen wir in dieser ersten Phase nmap nicht um Ports zu scannen sondern für sogenanntes „network mapping“. Hierbei werden „ICMP echo requests“ (auch als „ping“ bekannt) an alle Maschinen eines Zielnetzes geschickt. In unserem Fall wird jedoch ein TCP-Pingscan durchgeführt da wir nicht mit Administratorprivilegien ausgestattet sind. Aus den entsprechenden Antworten ergibt sich eine Liste der laufenden Rechner.

Wir führen dies exemplarisch am Netz 172.16.0.\* vor, wobei \* auch hier als Jokerzeichen fungiert und Werte von 0 bis 255 abdeckt. Der Parameter `-sP` leitet eben diese „pings“ ein.

```
bash-2.05a$ nmap -sP 172.16.0.*
```

```
Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Machine 172.16.0.2 MIGHT actually be listening on probe port 80
Host (172.16.0.2) appears to be up.
Host (172.16.0.3) appears to be up.
```

```
Nmap run completed -- 256 IP addresses (2 hosts up) scanned in 3 seconds
```

Weiters erhielten wir durch Anwendung dieses Befehls auf die anderen Netze (z.B. 192.168.0.\*) noch folgende Hosts:

```
Host GW-herr-der-ringe.student.com (192.168.0.5) appears to be up.
Host security.student.com (192.168.0.10) appears to be up.
Host grishnakh.fangorn.com (192.168.1.33) appears to be up.
Host ents.fangorn.com (192.168.1.35) appears to be up.
Host bambart.fangorn.com (192.168.1.57) appears to be up.
Host sam.mordor.com (192.168.2.2) appears to be up.
Host gollum.mordor.com (192.168.2.4) appears to be up.
Host frodo.mordor.com (192.168.2.17) appears to be up.
Host shelob.mordor.com (192.168.2.25) appears to be up.
Host celebron.lorien.com (192.168.3.1) appears to be up.
Host arwen.lorien.com (192.168.3.2) appears to be up.
Host haldir.lorien.com (192.168.3.99) appears to be up.
Host merry.hobbits.com (192.168.5.12) appears to be up.
Host pippin.hobbits.com (192.168.5.57) appears to be up.
```

```
Host gandalf.wizards.com (10.0.0.1) appears to be up.
Host saruman.wizards.com (10.0.0.2) appears to be up.
Host sauron.wizards.com (10.0.0.3) appears to be up.
Host galadriel.com.wizards.com (10.0.0.4) appears to be up.
Host (10.0.0.10) appears to be up.
Host radagast.wizards.com (10.0.0.100) appears to be up.
```

Wir haben es also mit vermeintlichen 22 Rechnern zu tun. Host und Domainname hat uns der Scan bereits geliefert. Natürlich interessiert uns an den gefundenen Rechnern auch welche Ports diese geöffnet haben was wir im nächsten Punkt eruieren.

## 1.2 Feststellung geöffneter Ports und des Betriebssystems:

Welche Ports auf einem untersuchten Rechner geöffnet sind ist ein sehr interessanter Punkt. Zum einen lässt sich eine gewisse Nutzungscharakteristik für den Rechner feststellen (File-server, DNS usw.) zum anderen kann aus eben diesen verwendeten Diensten oft sehr genau abgeleitet werden welches Betriebssystem sich auf dem Rechner befindet. Wobei wir hier anmerken möchten, dass es in anderen Konfigurationen von nmap (z.B. als „root“) auch möglich wäre explizit nach dem Betriebssystem zu scannen (OS-Fingerprint), jedoch wäre auch damit das Betriebssystem nicht 100 %ig zu ermitteln.

Die Vorgehensweise das Betriebssystem aus den laufenden(horchenden) Diensten zu ermitteln führen wir aus Gründen des Umfangs nur für drei Hosts exemplarisch durch, für alle restlichen Systeme sind die Ergebnisse der Scans sowie Betriebssystem in unserer Graphik „Netzwerktopologie“ angeführt.

Wir verwenden wieder nmap, wahlweise könnten wir auch noch mit dem Parameter `-p` den zu scannenden Portbereich angeben (z.B. 1025-2048) oder mit `-T` die Scangeschwindigkeit, wobei wir für unsere Scans den Standardportbereich und die „Normale“ Geschwindigkeit wählen.

```
bash-2.05a$ nmap 172.16.0.2
```

```
Interesting ports on (172.16.0.2):
(The 1551 ports scanned but not shown below are in state: closed)
Port      State      Service
21/tcp    open       ftp
25/tcp    open       smtp
80/tcp    open       http
```

Dieses Ergebnis ist nun noch nicht besonders charakteristisch für ein bestimmtes Betriebssystem. Wir bedienen uns des praktischen Tools nc (netcat) um mit den, durch den Scan herausgefundenen Diensten auf deren Port zu „sprechen“. Hierzu ein kurzes Beispiel:

```
bash-2.05a$ nc 172.16.0.2 25
220 localhost.localdomain ESMTP Sendmail 8.11.0/8.11.0; Thu, 13 Nov 2003
06:55:07 +0100
```

Da Sendmail grossteils nur auf Unix oder unixartigen Systemen betrieben wird, können wir hier davon ausgehen, dass es sich hier um einen Linux Rechner oder ähnliches handelt. Weiters lieferte ein Verbinden zu Port 80 noch genauere Auskünfte:

```
Server: Apache/1.3.12 (Unix) (Red Hat/Linux) mod_ssl/2.6.6 OpenSSL/0.9.5a
mod_perl/1.24
```

Ein weiteres Beispiel:

```
bash-2.05a$ nmap 192.168.2.4
```

```
Interesting ports on gollum.mordor.com ( ):
(The 1549 ports scanned but not shown below are in state: closed)
Port      State      Service
23/tcp    open       telnet
135/tcp   open       loc-srv
139/tcp   open       netbios-ssn
445/tcp   open       microsoft-ds
1025/tcp  open       listen
```

Auf Grund des offenen Ports 135 (Windows Netzwerk Freigaben) könne wir annehmen, dass es sich um einen Windows Rechner handelt. Recherchen auf einschlägigen Internetseiten[3] bestätigten diese Annahme.

```
bash-2.05a$ nmap 192.168.2.25
```

```
Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Interesting ports on shelob.mordor.com (192.168.2.25):
(The 1550 ports scanned but not shown below are in state: closed)
Port      State      Service
22/tcp    open       ssh
80/tcp    open       http
443/tcp   open       https
3128/tcp  open       squid-http
```

Hier bekamen wir beim Verbinden mit nc auf den „interessanten Ports“ unter anderem folgende Informationen:

```
bash-2.05a$ nc 192.168.2.25 80
get / http/1.0
Server: Apache/1.3.23 (Unix) (Red-Hat/Linux) mod_ssl/2.8.7 OpenSSL/0.9.6b
DAV/1.0.3 PHP/4.1.2 mod_perl/1.26
```

```
Weiters:
Server: Squid/2.4.STABLE6
```

Diese haben wir abermals als deutliches Signal für ein Red Hat Linux System angesehen.

Nun haben wir schon einige Informationen über unsere Nachbarsysteme herausgefunden aber um ein wirklich sinnvolles Netzwerkdiagramm erstellen zu können müssen wir auch wissen wie die von uns gefundenen Rechner untereinander bzw. mit unserem Startsystem verbunden sind.

### 1.3 Ermitteln der Netzwerktopologie

Um Hierarchie und Struktur unseres Netzwerks zu ermitteln bedienen wir uns des Unix-Programms traceroute eine ähnliche Variante ist unter Windows Systemen mit dem tracert-Befehl verfügbar. Uns blieb hier wenig anderes übrig als eben die Route die ein Datenpaket zu einem Rechner nimmt für jeden unserer ermittelten Rechner zu überprüfen. Wir werden unsere Informationsbeschaffung an ein paar kleinen Beispielen vorführen:

Wir ermitteln den Weg den ein Datenpaket zum Rechner shelob.mordor.com (192.168.2.25) nimmt.

```
bash-2.05a$ traceroute 192.168.2.25
traceroute to 192.168.2.25 (192.168.2.25), 30 hops max, 38 byte packets
 1  GW-herr-der-ringe.student.com (192.168.0.5)  1.047 ms  0.546 ms  0.287
ms
 2  radagast.wizards.com (10.0.0.100)  1.738 ms * *
 3  sauron.wizards.com (10.0.0.3)  7.033 ms  1.930 ms  1.229 ms
 4  shelob.mordor.com (192.168.2.25)  8.449 ms  1.497 ms  1.286 ms
```

Wir sehen, dass shelob.mordor.com (192.168.2.25) nur über die Hosts GW-herr-der-ringe.student.com, radagast.wizards.com und sauron.wizards.com erreichbar ist. Dies können wir bereits in unser Netzwerkdiagramm einzeichnen

Weiters dieselbe Vorgehensweise bei 192.168.1.33

```
bash-2.05a$ traceroute 192.168.1.33
traceroute to 192.168.1.33 (192.168.1.33), 30 hops max, 38 byte packets
 1  GW-herr-der-ringe.student.com (192.168.0.5)  1.561 ms  0.478 ms  0.398
ms
 2  * * *
 3  saruman.wizards.com (10.0.0.2)  6.402 ms  1.135 ms  4.693 ms
 4  grishnakh.fangorn.com (192.168.1.33)  2.234 ms  1.269 ms  1.064 ms
```

Hier haben wir wieder den Weg zu grishnakh.fangorn.com ermittelt und eingezeichnet

Für das 172.16.0.\* Netz möchten wir auch noch einen Test exemplarisch vorführen.

```
bash-2.05a$ traceroute 172.16.0.2
traceroute to 172.16.0.2 (172.16.0.2), 30 hops max, 38 byte packets
 1  GW-herr-der-ringe.student.com (192.168.0.5)  1.517 ms  1.611 ms  0.414
ms
 2  * * radagast.wizards.com (10.0.0.100)  1.559 ms
 3  galadriel.com.wizards.com (10.0.0.4)  2.361 ms  3.571 ms  1.301 ms
 4  arwen.lorien.com (192.168.3.2)  2.462 ms  1.666 ms  4.622 ms
 5  * * arwen.lorien.com (192.168.3.2)  1.878 ms
```

Hier sehen wir, dass uns ein Traceroute zu 172.16.0.2 eigentlich zum Rechner mit der IP-Adresse 192.168.3.2 geführt hat. Das heißt wir dürfen für das weitere Zeichnen des Diagramms nur mit Vorbehalt annehmen, dass es in dem Testnetzwerk wirklich die 22 anfangs gefundenen Rechner gibt, da durchaus ein Rechner zwei verschiedene IP-Adressen bzw. Rechnernamen haben kann. An dieser Stelle dürfen wir nun auf unser beigelegtes Netzwerkdiagramm verweisen, welches einerseits die durch Traceroute herausgefundenen Zusammenhänge zeigt aber auch unsere Vermutungen beinhaltet welche Rechner identisch sind. Hilfreich zur Feststellung welche Rechner wirklich im Netzwerk sind wäre uns natürlich das arp-Kommando gewesen, welches jedoch leider nicht verfügbar war.

## 2 Firewall-Scans

Testen der Durchlässigkeit einer Firewall.

### 2.1 Scan gegen 172.16.0.2

```
bash-2.05a$ firewall-sec 192.168.3.2 172.16.0.2
Firewalk Interface for Security VU
SET Modus = restricted DONE

Firewalk 5.0 [gateway ACL scanner]
Firewalk state initialization completed successfully.
TCP-based scan.
Ramping phase source port: 53, destination port: 33434
Hotfoot through 192.168.3.2 using 172.16.0.2 as a metric.
Ramping Phase:
 1 (TTL 1): expired [192.168.0.5]
 2 (TTL 2): expired [10.0.0.100]
 3 (TTL 3): expired [10.0.0.4]
 4 (TTL 4): expired [192.168.3.2]
Binding host reached.
Scan bound at 5 hops.

Scanning Phase:
...
  port 6: unknown (unreach ICMP_UNREACH_PORT) [192.168.3.2]
  port 7: *no response*
  port 21: A! open (port listen) [172.16.0.2]
  port 22: unknown (unreach ICMP_UNREACH_PORT) [192.168.3.2]
  port 23: A! open (port not listen) [172.16.0.2]
  port 24: *no response*
  port 25: A! open (port listen) [172.16.0.2]
  port 80: A! open (port listen) [172.16.0.2]
  port 220: A! open (port not listen) [172.16.0.2]
  port 389: A! open (port not listen) [172.16.0.2]
...
  port 1024: unknown (unreach ICMP_UNREACH_PORT) [192.168.3.2]

Scan completed successfully.

Total packets sent:          1028
Total packet errors:         0
Total packets caught         699
Total packets caught of interest 694
Total ports scanned          1024
Total ports open:            6
Total ports unknown:         684
```

Aus obigen Firewalk Scan können wir schließen das die Firewall vor dem System 172.16.0.2 die Ports: 21, 23, 25, 80, 220 und 389 durchlässt.

## 2.2 Scan gegen 172.16.0.3

```
bash-2.05a$ firewall-sec 192.168.3.2 172.16.0.3
```

```
Firewalk Interface for Security VU  
SET Modus = restricted DONE
```

```
Firewalk 5.0 [gateway ACL scanner]  
Firewalk state initialization completed successfully.  
TCP-based scan.  
Ramping phase source port: 53, destination port: 33434  
Hotfoot through 192.168.3.2 using 172.16.0.3 as a metric.  
Ramping Phase:  
 1 (TTL 1): expired [192.168.0.5]  
 2 (TTL 2): expired [10.0.0.100]  
 3 (TTL 3): expired [10.0.0.4]  
 4 (TTL 4): expired [192.168.3.2]  
   Binding host reached.  
   Scan bound at 5 hops.
```

```
Scanning Phase:
```

```
...
```

```
port 21: A! open (port not listen) [172.16.0.3]  
port 22: unknown (unreach ICMP_UNREACH_PORT) [192.168.3.2]  
port 23: A! open (port listen) [172.16.0.3]  
port 24: *no response*  
port 25: A! open (port listen) [172.16.0.3]  
port 80: A! open (port not listen) [172.16.0.3]  
port 220: A! open (port not listen) [172.16.0.3]  
port 1024: unknown (unreach ICMP_UNREACH_PORT) [192.168.3.2]
```

```
Scan completed successfully.
```

```
Total packets sent:          1028  
Total packet errors:         0  
Total packets caught         702  
Total packets caught of interest 695  
Total ports scanned          1024  
Total ports open:            6  
Totalportsunknown:
```

Aus obigen Firewalk Scan können wir schließen das die Firewall vor dem System 172.16.0.3 die Ports: 21, 23, 25, 80 und 220 durchlässt.

## Zusammenfassung

Trotz der Tatsache dass wir auf dem Testsystem, mit eher eingeschränkten Benutzerrechten ausgestattet waren und mit verhältnismäßig wenigen Werkzeugen auskommen mussten, konnten wir erstaunlich viele Informationen über den Aufbau des Netzwerks und die Konfiguration der Rechner herausfinden.

## Referenzen

- [1] Skoudis, Ed: Counter Hack, Prentice Hall 2002
- [2] Insecure-Homepage, [www.insecure.org/nmap](http://www.insecure.org/nmap), (Zugriff 13.11. 2003)
- [3] NTSecurity-Homepage, <http://ntsecurity.nu/papers/port445/>, (Zugriff 14.11.2003)