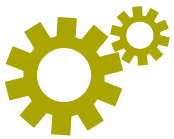
hakin9

Umgehung von Netzwerkfirewalls

Oliver Karow

Das *hakin9* Magazin, Wydawnictwo Software, ul. Piaskowa 3, 01-067 Warschau, Polen *de@hakin9.org*



Techniken

Umgehung von Netzwerkfirewalls

Oliver Karow

Schwierigkeitsgrad

Eine Firewall wird gerne als zuverlässiges Bollwerk zum Schutz von Netzwerken vor unauthorisierten Zugriffen eingesetzt. Jedoch können auch Firewalls Schwächen haben, die es einem Angreifer ermöglichen diesen Schutzmechanismus zu umgehen. Diese Schwächen können sowohl auf einer fehlerhaften Konfiguration als auch auf Fehlern im Firewallprodukt basieren. Im Rahmen dieses Artikels werden wir uns anschauen, welche Möglichkeiten für einen Angreifer bestehen eine Firewall zu umgehen.

Der Schutz eines Netzwerkes gegen Angriffe und ungewünschte Zugriffe aus nicht vertrauenswürdigen Netzwerken wie dem Internet ist eine der wichtigsten Anforderungen heutiger IT-Infrastrukturen. Gleichzeitig ist dies das klassische Einsatzgebiet von Netzwerkfirewalls. Die Hauptaufgabe einer Firewall ist die Separation von Netzwerken und die Entscheidung zu treffen, ob Pakete zwischen den Netzwerken weitergeleitet oder geblockt werden. Die zwei gängigsten Firewalltypen sind Paketfilter und Application-Layer Firewalls (siehe Kasten *Firewall Grundlagen*).

Unabhängig von der verwendeten Technologie benötigt eine Firewall eine Basis auf deren Grundlage sie entscheiden kann, ob ein Paket weitergeleitet oder geblockt wird. Diese Entscheidungsgrundlage ist die Firewallpolicy in Form von Access-Listen oder Filterregeln. Wir werden uns nachfolgend anschauen, welche Möglichkeiten es gibt eine Firewall zu umgehen, indem wir schlecht geschriebene Filterregeln, Schwächen in gängigen Protokollen sowie Einschränkungen der verschiedenen Firewalltypen ausnutzen.

Erkennung von Firewalls

Bevor man die verschiedenen Techniken eine Firewall zu umgehen anwendet, tut man gut daran sich erst einmal zu versichern, ob das zu attackierende System überhaupt von einer Firewall geschützt wird oder nicht. Die Existenz einer Firewall ist nicht immer so offensichtlich, wie es auf den ersten Blick erscheinen mag, da es einige Methoden gibt um eine Firewall mehr oder weniger unsichtbar zu machen. Da jedoch die Existenz einer Firewall die Ergebnisse einer

In diesem Artikel erfahren Sie...

- wie Firewalls funktionieren,
- wie man eine Firewall entdecken kann,
- wie man eine Firewall umgehen kann.

Was Sie vorher wissen/können sollten...

- die Funktionsweise von TCP/IPv4 verstehen,
- das ISO/OSI Referenzmodell verstehen.

Firewall Grundlagen

Eine Firewall ist grundsätzlich ein System welches den Übergang zwischen mehreren Netzwerken darstellt, und einen Filtermechanismus besitzt, der darüber entscheidet, ob ein Paket zwischen den angebundenen Netzwern weitergeleitet oder stattdessen blockiert wird. Eine Kategorisierung von Firewalls kann auf Basis des TCP/IP-Layers auf der die Analyse und Weiterleitung der Pakete erfolgt, kategorisiert werden:

Paketfilter

Paketfilter analysieren Pakete auf dem Netzwerk- (3) und dem Transportlayer (4) des ISO/OSI Modells. Demzufolge wertet ein Paketfilter die nachfolgenden Inhalte der Paketheader aus, um eine Filterentscheidung zu treffen:

- Protokoll (ICMP, OSPF, AH, ESP, *etc.*);
- Quell-IP-Adresse;
- Ziel-IP-Adresse;
- Quell-Port;
- Ziel-Port;
- TCP-Flags (SYN, ACK, RST, FIN, *etc.*).

Statefull/dynamische Paketfilter

Aufbauend auf den Features eines einfachen Paketfilters, überwacht und speichert ein statefull Paketfilter den Status jeder Verbindung in einer Statetable genannten Tabelle. So merkt sich die Firewall, wenn beispielsweise aus dem internen Netzwerk eine Verbindung nach außen aufgebaut wird, die zu der Verbindung gehörenden IP-Adressen und Port-Nummern. Antwortpakete werden nur dann in das interne Netzwerk zugelassen, wenn sie genau zu den in der Statetable gespeicherten Verbindungsdaten passen, und innerhalb eines definierten Zeitfensters erscheinen.

Des Weiteren sind einige Paketfilter dazu in der Lage dynamische Filterregeln zu erstellen. Hierzu wird die Kommunikation zwischen Client und Server beobachtet. Wird in dieser Kommunikation beispielsweise ein neuer TCP-/UDP-Port oder eine neue IP-Adresse ausgehandelt, die im Regelwerk der Firewall noch nicht freigeschaltet wurden, so erstellt die Firewall eine temporäre Regel, die den vereinbarten Port beziehungsweise die IP-Adresse für den Zeitraum der Kommunikation freischaltet. Applikationen, wie Oracle und Portmapper arbeiten unter anderem mit dynamischen Ports oder IP-Adressen.

Application Level Firewalls

Application Level Firewalls sind in der Lage Pakete bis einschließlich des Applikationslayers des ISO/OSI Modells zu analysieren. Neben den Features eines statefull/dynamischen Paketfilters sind sie somit fähig, den Payload eines Paketes zu inspizieren. Während Paketfilter ihre Filterentscheidungen nur anhand der Informationen der Paketheader treffen können, kann eine Application Level Firewall zusätzlich nach applikationsspezifischen Mustern suchen. So kann innerhalb einer HTTP-Verbindung zum Beispiel aufgrund von Kommandos wie `CONNECT` oder `DELETE` gefiltert werden, während ein Paketfilter grundsätzlich nur Verbindungen zu HTTP-Ports an sich zulassen oder unterbinden kann.

Application Level Firewalls benötigen für jedes Protokoll welches durch die Firewall bearbeitet werden soll, einen eigenen Proxy-Dienst. Da allerdings nicht für jedes Protokoll ein Proxy verfügbar ist, haben die meisten Hersteller ihre Produkte zusätzlich mit Paketfilterfähigkeiten und generischen Proxy-Diensten ausgestattet, die jedoch in der Regel nicht die Fähigkeit haben den applikationsspezifischen Payload auszuwerten.

Hybrid- und Layer-2 Firewalls

Viele Hersteller setzen mittlerweile auf Hybridtechnologie um das Beste von allen Firewalltypen zu bekommen. Das bedeutet, dass sie zum Beispiel Statefull Paketfilter sowie Application-Layer Proxydienste für gängige Protokolle einsetzen. Des Weiteren gibt es mittlerweile auch etliche Layer-2-Firewallsysteme auf dem Markt, jedoch sind diese noch nicht so weit verbreitet wie Paketfilter und Application-Layer Firewalls.

Attacke verfälschen beziehungsweise diese komplett unwirksam machen kann, ist es aus der Sicht eines Angreifers notwendig auf die Existenz einer Firewall zu testen. Schauen wir uns einige Techniken an, die man zur Erkennung von Firewalls anwenden kann.

Traceroute

Bei Traceroute handelt es sich um einen Mechanismus der dazu verwendet wird die Router aufzulisten, die ein Paket in einem Netzwerk auf dem Weg zum Ziel passieren muss. Handelt es sich bei einem der Router um eine Firewall, so besteht die Möglichkeit, dass wir deren IP-Adresse mit dieser Technik in Erfahrung bringen können.

Da Traceroute ein sehr alter Mechanismus ist, wird er von den meisten Firewalls geblockt. Es gibt jedoch immer noch einige Verständnisprobleme mit der Funktionsweise von Traceroute, die es einem Angreifer ermöglichen Traceroute durch eine Firewall hindurch auszuführen.

Listing 1 zeit das Ergebnis von traceroute, welches von einer Firewall geblockt wurde. Wie man sehen kann funktioniert der Mechanismus bis zu dem System mit der IP 10.4.4.254. Danach scheint ein weiteres System die traceroute Pakete zu blocken.

Schauen wir uns einmal an, wie Tracerouting funktioniert (siehe auch Abbildung 1). Um die Route eines IP-Paketes durch ein Netzwerk bestimmen zu können, wird das TTL-Feld des IP-Headers verwendet. Der Wert des Feldes wird jedes mal, wenn es einen neuen Router erreicht, um den Wert eins verringert. Ist der Restwert danach noch größer als null, wird das Paket an den nächsten Router weitergereicht. Dies wird solange weitergeführt, bis der Wert null erreicht wird. In diesem Fall verwirft der Router das Paket, und schickt eine Benachrichtigung darüber an den Absender.

Traceroute verwendet diesen Mechanismus, indem es das erste Paket mit einem TTL-Wert von 1 versendet. Da der erste Router bereits nach dem verringern des Wertes um 1 bereits den Wert 0 erreicht, wird er das Paket verwerfen und eine ICMP TTL-expired Nachricht zurücksenden. Damit hat traceroute den ersten Router ermittelt. Anschließend wird ein Paket mit einer TTL von 2 versendet. Dieses passiert den ersten

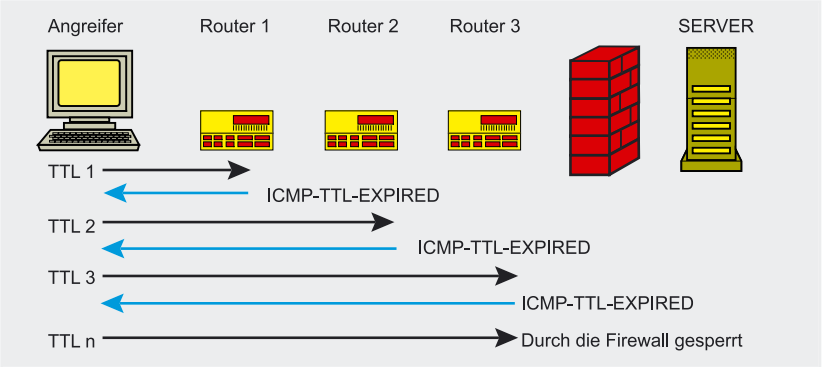


Abbildung 1. *Funktionsweise von traceroute*

Listing 1. *Von einer Firewall geblocktes Traceroute*

```
# traceroute www.dummycompany.de
traceroute to www.dummycompany.de (10.10.10.10), 30 hops max, 40 byte packets
 1  10.255.255.254                    0.373 ms  0.203 ms  0.215 ms
(...)
10  router.company1.de (10.1.1.254)  88.080 ms  88.319 ms  87.921 ms
11  router.company2.de (10.2.2.254)  87.881 ms  89.541 ms  88.081 ms
12  router.company3.de (10.3.3.254)  86.749 ms  86.919 ms  86.734 ms
13  router.company4.de (10.4.4.254)  87.216 ms  87.312 ms  87.307 ms
14  * * *
```

Listing 2. *TCP-Traceroute Technik mit hping2*

```
# hping2 -T -t 1 -S -p 80 www.dummycompany.de
HPING www.dummycompany.de (eth0 10.10.10.10 ): S set, ←
  40 headers + 0 data bytes
hop=1 TTL 0 during transit from ip=10.255.255.254 name=UNKNOWN
hop=1 hoprtt=12.4 ms
(...)
hop=10 TTL 0 during transit from ip=10.1.1.254 name=router.company1.de
hop=11 TTL 0 during transit from ip=10.2.2.254 name=router.company2.de
hop=12 TTL 0 during transit from ip=10.3.3.254 name=router.company3.de
hop=13 TTL 0 during transit from ip=10.4.4.254 name=router.company4.de
hop=14 TTL 0 during transit from ip=10.5.5.254 name=UNKNOWN
len=46 ip=10.10.10.10 flags=SA DF seq=15 ttl=107 id=12852 win=29200 rtt=95.6 ms
len=46 ip=10.10.10.10 flags=R DF seq=15 ttl=107 id=12856 win=0 rtt=194.6 ms
```

Listing 3. *Senden eines Paketes an einen geschlossenen Port*

```
# hping2 -S -p 99 -c 1 www.dontexist.com
HPING www.dontexist.com (eth0 192.168.10.10): S set, ←
  40 headers + 0 data bytes
ICMP Packet filtered from ip=192.168.9.254
```

Listing 4. *Beobachtung des Netzwerkverkehrs*

```
# tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
12:59:18.778417 IP 172.16.1.1.1866 > 192.168.10.10.99: ←
  S 1958445360:1958445360(0) win 512
12:59:18.786914 IP 192.168.9.254 > 172.16.1.1 icmp 36: ←
  host 192.168.10.10 unreachable - admin prohibited filter
```

Router, und erreicht am zweiten Router den Wert 0, womit dieser wieder dazu veranlasst wird, eine Nachricht zurückzusenden. Dieses Verfahren, den initialen TTL-Wert jeweils um eins zu erhöhen, wird solange durchgeführt bis das Zielsystem erreicht wurde. Da jeder Router eine solche Benachrichtigung versendet (sofern nicht anderweitig konfiguriert), kann traceroute sehr einfach eine Liste aller Router auf einem Pfad erstellen.

Es ist noch wichtig zu wissen, dass zwei verschiedene Implementierungen von traceroute existieren. Die eine Implementierung verwendet dabei ICMP *echo request* Pakete (bei Windows) und die andere UDP-Pakete (bei den meisten *NIX Implementierungen). Beide haben jedoch gemeinsam, daß sie die TTL-Technik verwenden. Für die Konfiguration einer Firewall ist es daher notwendig, beide traceroute Mechanismen zu filtern.

TCP traceroute

Da wir wissen, dass das TTL-Feld Teil des IP-Headers ist, und die üblichen Traceroute-Filter nur UDP- und ICMP-Pakete blocken, können wir versuchen den Filtermechanismus zu umgehen, indem wir TCP-Pakete anstelle von UDP- oder ICMP-Paketen verwenden. Versuchen wir nochmals den Pfad zu unserem Zielsystem zu tracen. Dieses mal verwenden wir das Tool *hping2,* das es uns ermöglicht TCP-Pakete mit modifizierten TTL-Werten zu versenden (siehe Listing 2). Wie wir sehen, konnten wir einen weiteren Router identifizieren. Während der *traceroute* Befehl nach dem dreizehnten Hop geblockt wurde, ging unser TCP-Traceroute bis zum Zielsystem durch, und zeigte mit dem Hop 14 einen weiteren Router auf.

Analyse von Antwortpaketen

Wenn man die Existenz einer Firewall ermitteln will, ist es ein guter Ansatz, die Antwortpakete von offenen Ports mit den Antwortpaketen von geschlossenen Ports zu vergleichen. Schauen wir uns ein paar Tricks an, die uns helfen

können die Existenz einer Firewall nachzuweisen.

Als erstes verwenden wir *hping2* um ein Paket an einen Port zu schicken, bei dem wir davon wissen oder davon ausgehen können, daß er geschlossen ist (siehe Listing 3). Zur gleichen Zeit lesen wir den Netzwerkverkehr mit *tcpdump* mit (siehe Listing 4). Wie wir sehen können, bekommen wir eine ICMP *destination unreachable* Mitteilung in Form einer *admin prohibited filter* Nachricht vom System 192.168.9.254. Die Nachricht zeigt, daß der Zugriff auf Port 99/TCP unseres Zielsystems mittels einer Access-Liste auf dem Router der die Nachricht gesendet hat, gefiltert wurde. Da dies ein sehr offensichtlicher Indikator für die Existenz einer Firewall ist, wollen wir uns noch eine andere Technik anschauen, die auf der Analyse der TTL-Werte basiert.

TTL Differenzen

Jedesmal wenn ein IP-Paket einen Router passiert, wird der TTL-Wert um eins reduziert. Wenn also vor einem Server eine Firewall platziert ist, besteht die Möglichkeit, dass Pakete die vom Server kommen einen unterschiedlichen TTL-Wert haben als die Pakete, die von der Firewall versendet werden.

Die Herausforderung besteht nun darin, sowohl den Server als auch die Firewall dazu zu bringen ein Antwortpaket an uns zu senden, damit wir nach Unterschieden im TTL-Feld schauen können. Unterscheiden sich die Felder mindestens um den Wert 1, so ist es sehr wahrscheinlich, daß ein Firewallsystem implementiert ist.

Um beide Systeme zum Antworten zu bringen, senden wir ein Paket an einen offenen und eines an einen geschlossenen Port, wobei 80/TCP offen und 99/TCP geschlossen ist (siehe Listing 5). Auf das Paket an den offenen Port sollte der Zielserver antworten, während auf das Paket an den geschlossenen Port die Firewall antworten sollte, sofern eine existiert. Wie wir sehen können unterscheiden sich die TTL's um den Wert 1. Das liegt daran, daß der Server ja hinter der Firewall steht, und somit dessen Pakete einen Router (hier die Firewall) mehr zu passieren haben als die Pakete die von der Firewall in unsere Richtung geschickt werden.

Listing 5. *Vergleich der TTL-Werte*

```
# hping2 -S -p 80 -c 1 www.randomname.com
HPING www.randomname.com (eth0 192.100.100.10): ←
  S set, 40 headers + 0 data bytes
len=46 ip=192.100.100.10 flags=SA DF seq=0 ttl=55 id=0 win=5840 rtt=7.6 ms

# hping2 -S -p 99 -c 1 www.randomname.com
HPING www.randomname.com (eth0 192.100.100.10): ←
  S set, 40 headers + 0 data bytes
len=46 ip=192.100.100.10 flags=RA DF seq=0 ttl=56 id=0 win=0 rtt=7.6 ms
```

Listing 6. *OS und Firewall Fingerprinting mit nmap*

```
# nmap -sS -F -n -O  -p 80,99,443 192.168.190.1
Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) ←
  at 2005-10-09 17:23 CEST
Interesting ports on 192.168.190.1:
PORT    STATE SERVICE
80/tcp  open  http
99/tcp closed metagram
443/tcp open  https
Device type: firewall|broadband router|general purpose
Running: Checkpoint Solaris 8, Belkin embedded, Sun Solaris 8
OS details: Checkpoint Firewall-1 NG on Sun Solaris 8, ←
  Belkin DSL/Cable Router, Sun Solaris 8, Sun Trusted Solaris 8
```

Listing 7. *Fingerprinting einer Symantec Enterprise Firewall*

```
Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) ←
  at 2005-10-10 13:43 CEST
Interesting ports on 192.168.99.1:
(The 1193 ports scanned but not shown below are in state: closed)
PORT     STATE    SERVICE
21/tcp   open     ftp
22/tcp   open     ssh
23/tcp   open     telnet
25/tcp   open     smtp
53/tcp   open     domain
80/tcp   open     http
119/tcp  open     nntp
139/tcp  open     netbios-ssn
443/tcp  open     https
481/tcp  open     dvs
512/tcp  open     exec
513/tcp  open     login
514/tcp  open     shell
554/tcp  open     rtsp
1720/tcp open     H.323/Q.931
2456/tcp open     unknown
5631/tcp open     pcanywheredata
7070/tcp open     realserver
No exact OS matches for host (If you know what OS is running ←
  on it, see http://www.insecure.org/cgi-bin/nmap-submit.cgi).
```

Bestimmen des Firewalltyps

Die vorher genannten Techniken helfen die Existenz einer Firewall nachzuweisen. Sofern wir dabei die IP-Adresse der Firewall herausfin-

Tabelle 1. *Standard Ports zur Bestimmung von Firewalls*

Firewall Produkt	Port Nummer	Zweck
Symantec Enterprise Firewall	888/TCP	OOB-Daemon
Symantec Enterprise Firewall	2456/TCP	webbasierte Administration
Checkpoint FW1-NG	256/TCP	Management
Checkpoint FW1-NG	257/TCP	FW1_log
Checkpoint FW1-NG	18181/TCP	OPSEC Content Vectoring Protocol
Checkpoint FW1-NG	18190/TCP	Management Interface

Listing 8. *Banner Checking*

```
# netcat www.raptorfirewall.nix 80
> HEAD / HTTP/1.0
< HTTP/1.1 503 Service Unavailable
< MIME-Version: 1.0
< Server: Simple, Secure Web Server 1.1
< Date: Fri, 17 Sep 2004 19:08:35 GMT
< Connection: close
< Content-Type: text/html
< <HTML>
< <HEAD><TITLE>Firewall Error: Service Unavailable</TITLE></HEAD>
```

den konnten, können wir nun noch ein paar weitere Tricks anwenden, um weiter Informationen wie das verwendete Firewallprodukt sowie das Betriebssystem zu bekommen.

TCP Fingerprinting

Wir benutzen die Tatsache das der IP-Stack eines Betriebssystems einzigartige Merkmale, vergleichbar mit einem Fingerabdruck, hat die es ermöglichen den Typ und die Version des Betriebssystems zu bestimmen. Da die meisten Firewallprodukte einen Einfluß auf das Verhalten des IP-Stacks haben, ist es auch oft möglich den Typ und die Version der verwendeten Firewall herauszufinden. Das Tool unserer Wahl zur Durchführung des Fingerprinting ist *nmap* mit seiner eingebauten OS-Erkennung (siehe Listing 6). Im nachfolgenden Beispiel haben wir lediglich drei Ports gescannt, und damit bereits herausgefunden, das es sich wahrscheinlich um eine Checkpoint Firewall-1 NG unter Sun Solaris handelt.

Schauen wir uns eine weitere Firewall an (siehe Listing 7). Dieses mal handelt es sich um eine Symantec Enterprise Firewall. Wie wir sehen können, war *nmap* nicht in der Lage Betriebssystem oder Firewallprodukt zu bestimmen. Aber die Tatsache, daß viele Ports offen sind, weist darauf hin, daß es sich um eine Proxy basierte Firewall handelt, wovon es jedoch nicht sehr viele Hersteller gibt.

Es lohnt sich also, zusätzlich zum Fingerprinting mit *nmap*, einen Blick auf die standard Ports von bekannten Firewallprodukten zu werfen. So hat beispielsweise die Symantec Enterprise Firewall (SEF) unter anderem die zwei typischen Ports 2456/TCP für die webbasierte Administration and 888/TCP für die Out-of-Band Authentisierung. Der Vergleich unseres Scanergebnisses mit Tabelle 1 bringt uns also auch wieder einen Schritt weiter Richtung Bestimmung des Firewallproduktes. Auch andere Firewallprodukte, wie beispielsweise die Checkpoint Firewall 1, verfügen über produkttypische Ports, wie die administrativen Ports in den Bereichen 256-264/TCP und 18180-18265/TCP.

Im Übrigen ist *nmap* nicht das einzige Tool das zum Fingerprinting von Firewalls verwendet werden kann. Die zusätzliche Verwendung von Programmen, wie *xprobe* und *p0f* kann die Erkennungsrate stark erhöhen. Weitere, nützliche Informationen über Fingerprinting können im Artikel *OS Fingerprinting – wir lassen uns nicht identifiezieren,* veröffentlicht in der Ausgabe 4/2004 von *hakin9*, nachgelesen werden.

Banner Checking

Um die Bestimmung des verwendeten Firewallproduktes weiterzubringen, empfiehlt es sich die Bannermeldungen bestimmter Firewalldienste anzuschauen. So kann zum Beispiel anhand der Ausgabe des HTTP-Daemons *Server: Simple, Secure Web Server 1.1* (siehe Listing 8) relativ einfach bestimmt werden, dass es sich um eine Symantec Enterprise Firewall handelt.

Allerdings ist die Bannermeldung alleine nicht sehr zuverlässig, da sie bei den meisten Daemons relativ einfach geändert werden kann. Aber in Verbindung mit dem Fingerprinting und den offenen Standardports stellt sie einen guten Indikator zur Betimmung des Firewallproduktes dar.

Umgehung von Firewalls

Nachdem ein Angreifer Existenz und Typ einer Firewall bestimmen konnte, bestehen mehrere Möglichkeiten diese zu umgehen. Nachfolgend wollen wir uns einige Methoden anschauen die zu einer Umgehung der Firewall führen können.

Quell-Port Angriffe

Beginnen wir mit einfachen Paketfiltern. Diese treffen ihre Entscheidungen indem sie die IP- und TCP/UDP-Header analysieren. Um einen Paketfilter so zu konfigurieren, das Benutzer eines Netzwerkes (intern) im Internet (extern) surfen können, benötigen wir eine Regel für die ausgehenden Pakete (HTTP-Anfragen des Browsers) sowie eine

Regel für die eingehenden Antwortpakete der Webserver. Um diese Regel erstellen zu können, müssen wir noch berücksichtigen, daß Webserver standardmässig auf dem Port 80/TCP eingerichtet sind, und der Quellport des Webbrowsers bei dessen Aufruf automatisch gewählt wird, und im Vorfeld nicht bekannt ist. Jedoch wird der Quellport per Definition größer 1024 sein. Tabelle 2 zeigt eine minimale Access-Liste für den Webverkehr.

Auf den ersten Blick sieht diese Regelwerk nicht sonderlich gefährlich aus. Regel 1 erlaubt ausgehende HTTP-Anfragen eines Webbrowsers und Regel 2 erlaubt die zugehörigen Antwortpakete des Webservers. Die dritte Regel blockiert alle Pakete die nicht zum erlaubten HTTP-Verkehr gehören, und wird daher Cleanup-Regel genannt. Eine solche Cleanup-Rule sollte in jedem Firewallregelwerk enthalten sein, ist aber für unser Beispiel nicht weiter relevant. Schaut man sich Regel 2 einmal etwas genauer an, so sieht man, daß ein Paket das aus dem Internet (Extern) stammt, und in das interne Netz (Intern) gesendet wird, dann der Regel entspricht, wenn es den Quellport 80 und einen Zielport größer 1024 hat.

Einen Angriff, der diesen Sachverhalt ausnutzt wird *High-Port* oder *Source-Port* Angriff genannt, da er auf der Tatsache basiert, dass ein Angreifer nur den Quellport seiner Hackertools auf einen Wellknown-Port, wie in unserem Beispiel der Port 80/TCP, einstellen muss, und damit Server hinter der Firewall angreifen kann, sofern diese auf einem High-Port lauschen. Einige interessante Dienste, die einen High-Port verwenden, sind Xwindow (6000-6063/TCP), Windows Terminal Server (3389/TCP) sowie viele Webapplikationen wie Jakarta Tomcat (8080/TCP) und Bea Weblogic (7001/TCP).

Listing 9. *Normaler Scan versus Sourceport-Scan*

```
# nmap -sS -p 1-65535 192.168.0.1
Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) ←
  at 2005-10-09 17:01 CEST
Interesting ports on 192.168.0.1:
(The 1658 ports scanned but not shown below are in state: closed)
PORT   STATE SERVICE
80/tcp open  http
Nmap run completed -- 1 IP address (1 host up) scanned in 6.607 seconds

# nmap -sS -g 80 -p 1024-65535 192.168.0.1
Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) ←
  at 2005-10-09 17:01 CEST
Interesting ports on 192.168.0.1:
(The 1657 ports scanned but not shown below are in state: closed)
PORT   STATE SERVICE
80/tcp open  http
6000/tcp open X11
Nmap run completed -- 1 IP address (1 host up) scanned in 6.607 seconds
```

Um zu testen, ob eine Firewall verwundbar für diese Angriffe ist, kann *nmap* mit der `-g` Option verwendet werden, die dafür sorgt das man den Quellport festlegen kann. Listing 9 zeigt den Unterschied zwischen einem normalen und einem Sourceport Scan.

Wie man sehen kann, konnte ein weiterer offener Port (6000/TCP) mittels des Sourceport Scans gefunden werden. Damit ein Angreifer sich über die Firewall hinweg mit dem Dienst verbinden kann, muss er nur noch seine Tools so konfigurieren, dass sie den Quellport 80/TCP verwenden.

Eine einfache Methode eine TCP-Verbindung mit einem festen Quellport aufzubauen, ist die Verwendung eines Tools wie FPipe von Foundstone. FPipe ist zwar ein Windows tool, kann unter Linux jedoch auch mittels Wine verwendet werden. Der Aufruf mit den folgenden Optionen:

```
> FPipe -l 100 -s 80 -r 6000 ←
  192.168.0.1
```

öffenet einen Listener auf dem lokalen Port 100/TCP. Alle Pakete die an diesen lokalen Port gesendet werden, werden automatisch an den Zielport 6000/TCP der IP-Adresse 192.168.0.1 weitergeleitet, und mit dem Quellport 80/TCP versehen.

Bei der Prüfung, ob eine Firewall auf Sourceport-Angriffe anfällig ist, sollte man zusätzlich mit den Quellports 53 (DNS), 20 (FTP) und 88 (Kerberos) testen, da einige Firewalls aufgrund der besonderen Eigenheiten dieser Protokolle, auch solche Pakete durchlassen. Als Beispiel sei hier die Checkpoint FW1 genannt, die bis einschließlich der Version 4.1 über sogenannte *Implied Rules* verfügte, die DNS-Pakete aus jedem Netz durchließen.

Auch Microsofts Implementierung des IPSec Filters, der als lokale

Tabelle 2. *Vereinfacht dargestellte Access-Liste für HTTP-Verkehr*

Nr.	Quell-IP	Ziel-IP	Quell-Port	Ziel-Port	Aktion	Beschreibung
1.	Intern	Extern	>1024/TCP	80/TCP	Allow	Erlaube ausgehende HTTP Anfragen
2.	Extern	Intern	80/TCP	>1024/TCP	Allow	Erlaube HTTP-Antworten
3.	Alle	Alle	Alle	Alle	Deny	Cleanup rule

Tabelle 3. *Vereinfacht dargestelltes Regelwerk einer statefull Firewall*

Nr.	Quell-IP	Ziel-IP	Quell-Port	Ziel-Port	ACK-Flag erforderlich	Aktion	Beschreibung
1.	Intern	Extern	>1024/TCP	80/TCP	Nein	Allow	Erlaube ausgehende HTTP Anfragen
2.	Extern	Intern	80/TCP	>1024/TCP	Ja	Allow	Erlaube HTTP-Antworten
3.	Alle	Alle	Alle	Alle	–	Deny	Cleanup rule

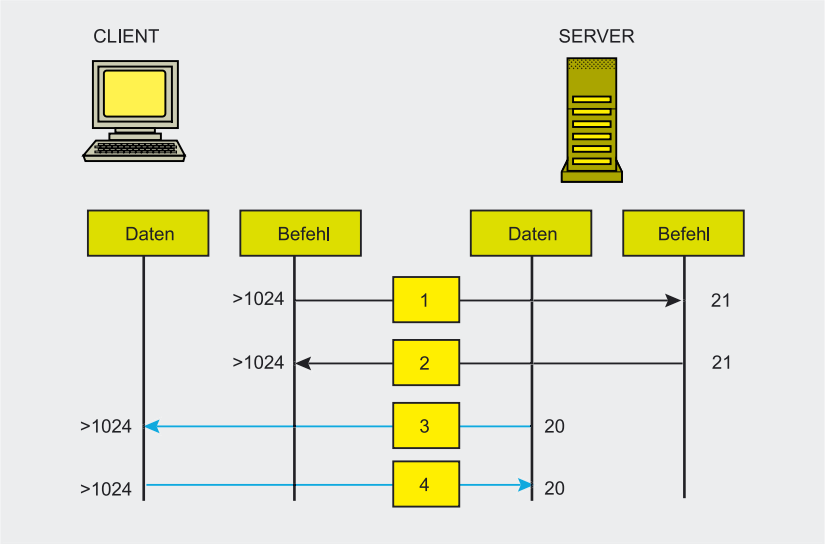


Abbildung 2. *Wie Aktiv-FTP funktioniert*

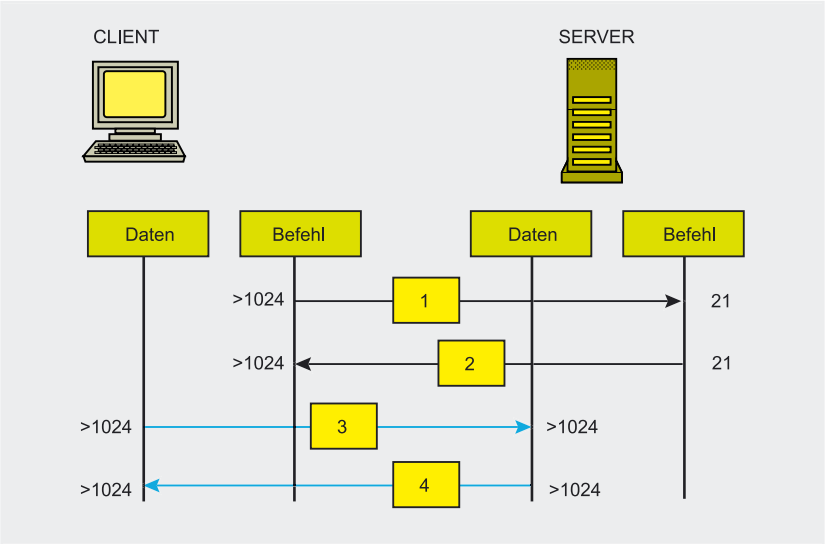


Abbildung 3. *Wie Passiv-FTP funktioniert*

Firewall konfiguriert werden kann, hat eine ähnliche Schwachstelle. Eine hartcodierte, unsichtbare Regel lässt jedes Paket passieren, sofern es den Quellport 88 (Kerberos) hatt. Um dies zu unterbinden, müssen Änderungen an der Registry vorgenommen werden.

Statefull Firewalls

Um einen Angreifer davon abzuhalten Verbindungen zu internen Systemen aufbauen zu können indem er einfach die Antwort eines Servers simuliert, ist es für eine Firewall wichtig zwischen einem Antwortpaket und einem Paket, das eine neue Verbindung initiiert, unterscheiden zu können. Dazu kann eine Firewall die unterschiedlichen Flags innerhalb des TCP-Headers auswerten. Da jede neu zu initiierende TCP-Verbindung mit einem gesetzten SYN-Flag beginnt, und alle darauffolgenden Pakete in der Verbindung mindestens das ACK-Flag gesetzt haben, gibt es ein eindeutiges Attribut für eine Firewall um die Unterscheidung machen zu können. Zusätzlich hilft die interne Statetable der Firewall, sich den Zustand der Verbindungen zu merken. Dies ist besonders bei UDP-Kommunikation, in der es keine Status-Flags gibt, hilfreich.

Wie wir Tabelle 3 entnehmen können, wird das Antwortpaket eines HTTP-Servers nur dann durchgelassen, wenn im TCP-Header das ACK-Flag gesetzt ist. In diesem Fall funktioniert die Sourceport-Attacke nicht mehr, da TCP-Verbindungsaufbauten immer mit einem SYN-Paket beginnen. Der Angreifer muss sich somit nach einer anderen Möglichkeit umsehen die Firewall zu umgehen.

Ausnutzen von Aktiv-FTP

Eines der meist verwendeten Protokolle im Internet ist das *File Transfer Protocol* (FTP). Es existieren zwei verschiedene Arbeitsweisen von FTP, der Aktiv- und der Passivmodus (siehe Kasten *FTP Aktiv- und Passivmodus*). Der Hauptunterschied zwischen den beiden Modi ist die Art und Weise, wie die Kommunikation aufgebaut wird. Während im Aktivmodus der FTP-Client den Kommando-Kanal aufbaut, und der Server für den Datenkanal eine Verbindung zum Client herstellt, werden im Pas-

FTP Aktiv- und Passivmodus

Das *File Transfer Protocol* verwendet zwei Kanäle für die Kommunikation zwischen Client und Server. Der Kommandokanal wird verwendet um Befehle an den Server zu senden, und die Antworten des Servers zu empfangen. Sobald Daten übertragen werden, wird ein dedizierter Datenkanal aufgebaut. Dies ist beispielsweise der Fall, wenn eine Datei hoch- oder heruntergeladen wird, aber auch dann wenn ein Liste der Dateien und Verzeichnisse mit einem Befehl wie `dir` angefordert wird.

Für den Aufbau des Datenkanals unterstützt FTP zwei unterschiedliche Modi, den Aktiv-und den Passiv-Modus. Der Unterschied zwischen den beiden Modi liegt darin, ob der Client oder der Server den Aufbau des Datenkanals initiiert.

Im Falle von Aktiv-FTP baut der FTP-Server die Verbindung zum FTP-Client auf. Hierzu teilt der FTP-Client dem Server mittels des `PORT` Kommandos mit, an welcher IP-Adresse und auf welchem Port er einen Listener errichtet und Verbindungsanfragen des Servers entgegennimmt. Im Falle von Passiv-FTP baut der Client die Verbindung auf. Hierzu teilt der FTP-Server dem Client mit, an welcher IP-Adresse und an welchem Port der Server die Verbindung entgegen nimmt.

Um in den Passiv-Modus zu gelangen muss der Client dem Server das `PASV` Kommando senden. Als Antwort darauf sendet der Server die Socketinformation im Format `IP,IP,IP,IP,Hbyte,Lbyte` wobei *Hbyte* und *Lbyte* den Port darstellen, und die einzelnen Bytes der IP-Adresse durch Kommas anstelle von Punkten getrennt sind. Siehe auch Abbildungen 2 und 3.

sivmodus beide Verbindungen vom Client aufgebaut.

Der Angriff gegen Aktiv-FTP ist eine weitere Art eines Source-Port-Angriffs. Jedoch kann dieser Angriff nicht durch die Filterung anhand der TCP-Flags verhindert werden, da die Verwendung von Aktiv-FTP eingehende Verbindungsanfragen mit gesetztem SYN-Flag für den Aufbau des Datenkanals benötigt (siehe Tabelle 4 für ein Beispielregelwerk). Um eine Verbindung zu internen Diensten die auf Highports laufen, aufbauen zu können, muss der Angreifer lediglich den Quellport seiner Tools auf 20/TCP festlegen.

Um zu prüfen ob die Firewall verwundbar für diese Art von Angriffen ist, können wir erneut *nmap* verwenden, jedoch diesmal mit der Option `-g 20` anstelle von `-g 80`. Um eine richtige TCP-Verbindung aufbauen zu können, kann FPipe wie im vorherigen Beispiel verwendet werden.

Ausnutzen von Passiv-FTP

Die meisten aktuellen FTP-Server unterstützen heutzutage den Passivmodus, aber leider gibt es noch einige FTP-Clients (darunter der standard Microsoft FTP-Client) die nur den Aktivmodus unterstützen. Unabhängig davon gibt es jedoch auch Angriffe gegen Passiv-FTP, die die Umgehung einer Firewall ermöglichen. Schauen wir uns einmal die Kommunikation im Passivemodus an. Um die Lesbarkeit zu erhöhen verwenden wir *netcat* zur Erstellung der Verbindung (siehe Listing 10).

Die ersten sechs Zeilen sind die Standardkommunikation zwischen Client und Server und dienen der Anmeldung am FTP-Server. In zeile sieben wird dem FTP-Server mitgeteilt, dass für die Erstellung des Datenkanals der Passivmodus verwendet werden soll.

Als Antwort auf diese Anforderung teilt uns der Server in Zeile acht die IP-Adresse und den Port mit, die er für die Erstellung des Datenkanals öffnet, damit sich der Client dorthin verbinden kann.

Sofern eine Firewall vor dem FTP-Server steht, weiß diese erstmal nicht welcher Port für den Datenkanal vereinbart wurde. Daher hat die Firewall grundsätzlich zwei Möglichkeiten um die FTP-Kommunikation zu ermöglichen:

- die erste Option besteht darin alle Highports für eingehende Verbindungsanfragen an den FTP-Server durchzulassen. Dies ist sehr unsicher, insbesondere wenn auf dem FTP-Server noch weitere Dienste auf Highports laufen, oder sich eine Vielzahl von FTP-Servern im internen Netzwerk befinden. Wir gehen daher davon aus, daß die Firewall in unserem Beispiel von dieser Konfiguration keinen Gebrauch macht;
- die zweite Option besteht darin, daß die Fiewall die Kommunikation zwischen Client und Server im Kommando-Kanal analysiert. Sieht die Firewall ein Kommando im Format `227 Entering Passive`

Tabelle 4. *Regelwerk für Aktiv FTP*

Nr.	Quell-IP	Ziel-IP	Quell-Port	Ziel-Port	ACK-Flag erforderlich	Aktion	Beschreibung
1.	Intern	Extern	>1024/TCP	21/TCP	Nein	Allow	Command channel
2.	Extern	Intern	21/TCP	>1024/TCP	Ja	Allow	Command channel
3.	Extern	Intern	20/TCP	>1024/TCP	Nein	Allow	Data channel
4.	Intern	Extern	>1024/TCP	20/TCP	Ja	Allow	Data channel
5.	Any	Any	Any	Any	–	Deny	Cleanup rule

Listing 10. *Passiv-FTP Kommunication*

```
# nc ftp.hakin9.org 21
< 220-Welcome to hakin9.org.
> user anonymous
< 331 Please specify the password.
> pass secret
< 230 Login successful.
> pasv
< 227 Entering Passive Mode (192,168,200,23,230,242)
```

Listing 11. *Öffnen eines Ports mittels Passiv-FTP*

```
# nc ftp.hakin9.org 21
< 220-Welcome to hakin9.org.
> user anonymous
< 331 Please specify the password.
> pass secret
< 230 Login successful.
> AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA227 ↩
  Entering Passive Mode (192,168,200,23,0,2)
< 500 command not understood: ↩
  'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA227 ↩
  Entering Passive Mode (192,168,200,23,0,22)'
```

`Mode (IP,IP,IP,IP,Hbyte,Lbyte)`, das vom Server zum Client gesendet wird, so erstellt sie eine temporäre Regel, die eine eingehende Verbindung auf die vereinbarte IP/Port-Kombination zulässt.

Agiert die Firewall nach der zweiten Methode, so besteht die Möglichkeit das ein Angreifer sie dazu bringen kann einen beliebigen Port und gegebenenfalls auch eine beliebige IP-Adresse freizuschalten. Hierzu muss er den FTP-Server dazu bringen ein Paket an den Client innerhalb des Kommand-Kanals zu schicken, das dem genannten Format entspricht, und den Port und die IP-Adresse, auf die er sich verbinden möchte, enthält. Dies kann der Angreifer zum Beispiel erreichen, indem er eine Fehlermeldung erzwingt, die genau diese Daten enthält.

Sendet man ein nicht existierendes Kommando wie *AAAAAAAAAAA227 Entering Passive Mode 1,2,3,4,0,22* an einen FTP-Server, so sendet dieser in vielen Fällen eine Fehlermeldung zurück, die das fehlerhafte Kommando anzeigt: *cannot understand command AAAAAAAAAAA227 Entering Passive Mode 1,2,3,4,0,22.*

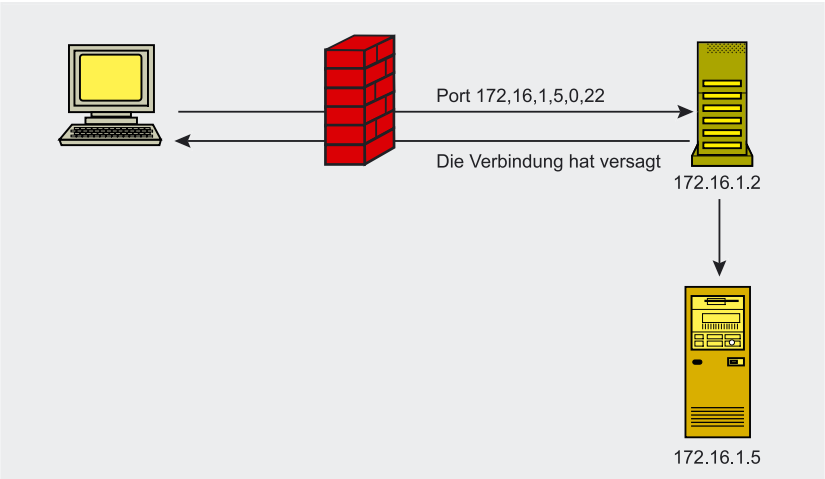


Abbildung 4. *Funktionsweise von FTP Bounce Scanning*

Berechnet man die Größe der Fehlermeldung derart, dass die Nachricht zu groß ist für ein IP-Paket, und dadurch das nicht existierende Kommando und die Passive Mode Zeichenkette in getrennten Paketen enthalten sind, kann man die Firewall dazu bringen die Nachricht fehlzuinterpretieren, und eine Verbindung zu Port 22/TCP der IP 1.2.3.4 zuzulassen.

Die Firewall wird die Pakete mit den `A` Zeichen unbeachtet passieren lassen.Wenn sie jedoch den Text `227 Entering Passive Mode (192,168,200,23,0,22)` erhält, wird sie eine temporäre Regel erstellen die es dem FTP-Client erlaubt sich auf Port 22 des Systems 192.168.200.23 zu verbinden. Ein ähnlicher Mechanismus der Erstellung von dynamischen Filterregeln wird übrigens auch für andere Protokolle, wie Oracle's SQLnet verwendet.

FTP Bounce Scanning

FTP Bounce Scanning (siehe Abbildung 4) nutzt ein Feature von Aktiv-FTP aus, um Systeme hinter einer Firewall auf offene Ports abscannen zu können. Innerhalb einer Aktiv-FTP Sitzung wird der Datenkanal vom FTP-Server aufgebaut, indem er sich auf einen offenen Port des FTP-Clients verbindet. Hierzu muss der Client dem Server innerhalb des Kommando-Kanals mitteilen, wohin er sich verbinden darf, da der Port auf dem Client zufällig ausgewählt wird.

Diese Information wird dem Server mittels des `PORT` Kommandos mitgeteilt. Ähnlich dem `PASV` Kommando ist der Syntax `PORT IP,IP,IP,IP,Hbyte,Lbyte` wie zum Beispiel `PORT 192,168,100,10,0,123`. Mit dieser Information ist der Server nun in der Lage eine Verbindung zu 192.168.100.10:123 aufzubauen, wenn Daten übertragen werden müssen. Per Definition besteht keine Einschränkung, daß es sich bei der übermittelten IP-Adresse um

Fragmentierung

Jedes Betriebssystem versucht die maximale Größe eines IP-Paketes an die maximale Größe eines Frames der darunterliegenden Layer2-Technologie anzupassen. So beträgt für Ethernet beispielsweise die maximale Größe 1518 Bytes. Da ein Ethernet-Frame selbst 18 Byte für seine Headerinforamation benötigt, bleibt für das zu transportierende IP-Paket ein Platz von 1500 Byte. Die sich hierdurch ergebende Größe wird *Maximum Transfer Unit* (MTU) genannt.

Auf seinem Weg über ein Netzwerk kann ein IP-Paket Router passieren, deren MTU aufgrund der verwendeten Layer-2-Technologie kleiner 1500 Byte ist. In einem solchen Fall sind die zu großen Pakete in mehrere kleinere Pakete zu zerteilen. Dieser Vorgang wird Fragmentierung genannt.

Am anderen Ende der Kommunikation sammelt der Zielserver die IP-Fragmente und fügt sie in der richtigen Reihenfolge wieder zusammen. Dieser Vorgang wird Reassemblierung genannt. Die Reassemblierung benötigt einige zusätzliche Informationen, um die Pakete in der richtigen Reihenfolge zusammenzufügen, und nicht mit Fragmenten aus anderen Verbindungen zu vermischen. Hierzu dienen die zwei Felder *Fragment Offset* und *Identification* (ID) innerhalb des IPv4-Headers. Jedes Fragment des gleichen Datagrams hat die gleiche ID. Dies ermöglicht es dem IP-Stack zu erkennen welche Fragmente zum gleichen Datagram gehören. Um die Pakete dann auch noch in der richtigen Reihenfolge zusammensetzen zu können, wird der *Fragment Offset* verwendet. Das erste Fragment hat stets einen Offset von null. Der Offset eines jeden folgenden Fragments erhöht sich um den Wert der Länge des Datenteils des Fragments. Das *More fragments* Bit (MF) im IP-Header zeigt an, ob noch weitere Fragmente folgen, oder ob es sich um das letzte Fragment handelt.

die IP-Adresse des selben Systems handeln muss, das auch den Kommando-Kanal aufgebaut hat (also der FTP-Client). Stattdessen nehmen einige FTP-Server auch andere IP-Adressen an.

Nachdem der Client einen Befehl wie `dir` an den Server geschickt hat, versucht der Server eine Verbindung zu dem vorher vereinbarten IP:port herzustellen, um die Daten zu übertragen. Abhängig davon, ob der Port offen oder geschlossen ist, gibt der Server einen Erfolgs- oder Fehlercode zurück an den Client. Analysiert man diese Meldungen, hat man einen Portscanning Mechanismus. Um dieses Vorgehen zu automatisieren kann *nmap* verwendet werden, da es FTP Bounce Scanning unterstützt:

```
$ nmap -b \
  anonymous@myftpserver:21 \
  targetserver
```

HTTP Proxy Bouncing

Applikationsfirewalls arbeiten in der Regel als HTTP-Proxies, entweder transparent oder nichttransparent, um HTTP-Verkehr filtern zu können. Ein Problem besteht dann, wenn ein HTTP-Proxy schlecht konfiguriert ist, und dadurch Zugriff von außen auf interne Systeme zulässt.

Der einfachste Weg festzustellen, ob eine Firewall verwundbar für Proxy-Bouncing ist, besteht darin, daß externe Interface der Firewall als Proxy für den eigenen Webbrowser einzutragen, und zu versuchen auf internen Systemen zu surfen:
Einstellungen für Lynx:

```
# http_proxy='http://myfirewall.de:8080'
# no_proxy='localhost'
# export http_proxy no_proxy
```

Surfen auf interne Webseiten:

```
# lynx 192.168.100.20
```

Ein schöner Nebeneffekt dieser Technik ist, daß man sogar private IP-Adressen hinter der Firewall über das Internet erreichen kann, da der Angreifer sich direkt mit der Firewall verbindet, und der darauf befindliche HTTP-Daemon die Verbindung zu den internen Systemen aufbaut. Da die Firewall die IP-Adressen der internen Systeme kennt, und diese auch erreichen kann, wenn sie privat (als nicht über das Internet routbar) sind, ist es für einen Angreifer möglich auf den internen Webservern zu surfen.

Wenn man schon mal dabei ist, sollte mal im gleichen Test auch versuchen andere Ports auf den internen Systemen zu erreichen:

```
# lynx 192.168.100.20:25
```

Man sollte jedoch darauf achten, daß einige Browser wie Mozilla Firefox und Lynx in der Standardeinstellung solche Requests blocken. Es ist daher empfohlen, solche Tests mit Netcat oder Telnet durchzuführen.

HTTP-Connect

Die HTTP-Option `CONNECT` dient normalerweise dazu SSL-Kommunikation durch einen Proxy tunneln zu können. Hierzu erstellt der Proxy lediglich eine TCP-Verbindung zu dem Zielserver und leitet die Pakete des Clients weiter. Leider prüfen einige Firewalls nicht, ob die IP-Adresse oder der Port der mit dem `CONNECT` Befehl übergeben wird legitim ist, und öffnen dadurch große Löcher für einen Angreifer.

Wie bekannt ist, sollten Firewalls so installiert werden, dass die administrativen Ports nur an den internen Interfaces erreichbar sind. Dies verhindert, dass ein Angreifer Attacken wie Buffer Overflows gegen den Administrationsdienst fahren, oder Anmeldedaten erraten kann.

Die `CONNECT` Schwachstelle erlaubt einem Angreifer jedoch eine Verbindung zu den administrativen Ports über ein externes Interface der Firewall aufzubauen:

```
# nc firewall 8080
CONNECT 127.0.0.1:22 HTTP/1.0
SSH-1.99-OpenSSH_3.8p1
```

Mittels der `CONNECT` Methode kann ein Angreifer auch Verbindungen zu anderen Systemen als der Firewall selbst aufbauen. Analog zur Proxy Bouncing Attacke können hiermit auch interne Systeme mit privaten IP-Adressen erreicht werden:

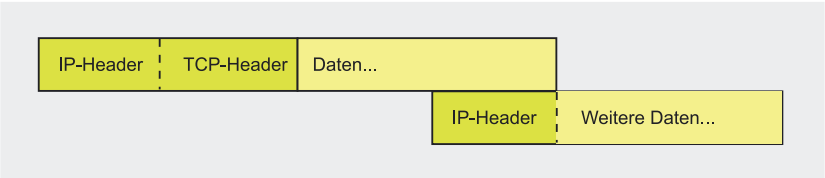


Abbildung 5. *Normale Reassemblierung eines TCP-Packets*

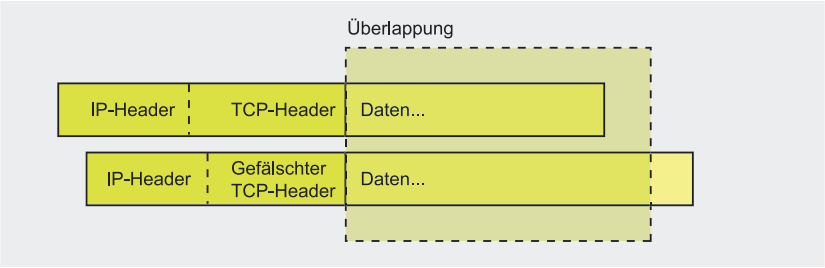


Abbildung 6. *Overlapping Fragment Attack – Überlappender Header*

```
# nc firewall 8080
CONNECT 10.1.1.100:25 HTTP/1.0
220 mailserver ESMTP
```

Wie wir sehen ist es sehr einfach, eine Firewall auf CONNECT Verwundbarkeiten zu testen. Die gleiche Technik kann man auch, ähnlich des FTP-Bounce-Scanning, dazu verwenden über eine Firewall hinweg interne Systeme abzuscannen. Interessanterweise waren auch bekannte Firewallprodukte wie Checkpoint FW1 und Astaro Secure Linux in der Vergangenheit bereits anfällig für solche Angriffe.

Overlapping Fragment Angriff

Das Ziel eines Overlapping Fragment Angriffes ist es Informationen im UDP oder TCP Header zu überschreiben, nachdem die Firewall bereits eine positive Entscheidung anhand des ersten Fragments getroffen hat. Wenn eine Fragmentierung von IP-Paketen stattfindet, so enthält lediglich das erste Fragment den TCP/UDP-Header mit den zugehörigen Informationen wie dem Zielport der Verbindung.

Besteht beispielsweise eine Firewallregel, die Verbindungen auf Port 80/TCP eines Webservers zulässt und gleichzeitig Verbindungen auf den Secure Shell Daemon des gleichen Servers auf Port 22/TCP unterbindet, lohnt es sich eine Overlapping Fragment Attacke durchzuführen.

Um die genannte Filterregel umgehen, und eine Verbindung auf Port 22 aufbauen zu können, kann ein Angreifer die IP-Datagramme fragmentieren (siehe Kasten *Fragmentierung*) und den Zielport innerhalb des TCP-Headers auf 80 setzen. Das Fragment erreicht die Firewall und wird aufgrund der Regel erlaubt. Da alle Fragmente die zum selben Datagram gehören eine identische IP-ID besitzen, wird die Firewall alle Fragmente durchlassen, die die gleiche IP-ID sowie Quell- und Ziel-IP wie das erste Fragment haben.

Nehmen wir an, dass der Offset des ersten Fragments null ist, und das Ende des Fragments bei Byte 128 liegt. Der Offset des zweiten Fragments sollte nun einen Wert aufweisen der direkt nach dem Byte 128 anknüpft. Ist der Offset jedoch kleiner als 128, so werden Bereiche des ersten Fragmentes überschrieben. Einen solchen Offset nennt man negativen Offset. Berechnet der Angreifer den Offset so, daß er den Zielport im TCP-Header des ersten Fragments überschreibt, ist es möglich den Wert 80 mit 22 zu überschreiben (siehe Abbildungen 5 und 6).

Nach der vollständigen Reassemblierung, entweder auf der Firewall selbst oder am Zielsystem, wird eine Verbindung zu Port 22/TCP anstelle von 80/TCP hergestellt und die Firewall wurde erfolgreich umgangen.

Es existieren unterschiedliche Implementierungen von Fragmentierungsangriffen gegen Firewalls. Ein Link zu einer interessanten Variante gegen BSD's IPFilter ist im Kasten *Im Internet* zu finden.

Tunneling Angriffe

Gelegentlich möchten Hacker durch eine Firewall hindurch mit internen Systemen kommunizieren, um zum Beispiel mit Rechnern die mit Trojanischen Pferden oder Backdoors infiziert sind, in Verbindung bleiben zu können. In einem solchen Fall sendet ein Hacker mittels eines beliebigen Protokolls Befehle an einen infizierten Rechner, und bekommt die Ergebnisse als Antwort zurück gesendet.

Wenn Filterregeln auf der Firewall jedoch nur die üblichen Protokolle wie HTTP, FTP und DNS für ausgehenden Verkehr zulassen, muss der Angreifer auch eines dieser Protokolle für seine Kommunikation verwenden. Unglücklicherweise (für den Hacker) prüfen einige moderne Firewalls den applikationsspezifischen Verkehr auf Einhaltung der RFC-Vorgaben, und blocken Datenpakete die nicht diesen Vorgaben entsprechen.

Angreifer, die das wissen, verwenden einen Tunnelingmechanismus, der bei der Kommunikation nicht gegen RFC-Vorgaben verstösst, indem die zu transportierenden Daten in gültige, protokollkonforme Befehle eingebettet werden. Wenn diese eingebetteten Daten zusätzlich kodiert oder verschlüsselt und ausschliesslich unter Verwendung von 7-bit ASCII-Zeichen versendet werden, ist eine Erkennung des Tunneling durch die Firewall beinahe unmöglich.

Gute Implementierungen hierfür sind HTTP- und DNS-basierte Tunnel. Während Tools für ein RFC-konformes HTTP-Tunneling wie rwwwshell (siehe Kasten *Im Internet*) relativ einfach umzusetzen, und daher auch schon seit etlichen Jahren verfügbar sind, sind gute DNS basierte Tunnel etwas aufwendiger.

Über den Autor

Oliver Karow arbeitet als Principal Security Consultant für einen Hard- und Softwarehersteller. Seine derzeitigen Schwerpunkte liegen auf der Implementierung von Firewall-, IDS/IPS-Systemen sowie der Durchführung von Sicherheitsaudits und Penetrationstests. Oliver Karow studiert nebenberuflich Informatik an einer deutschen Fernuniversität. Er arbeitet in der IT seit 1994, and seit 1999 mit Fokus auf IT-Sicherheit.

Ein interessanter DNS-Tunnel, der neben anderen eine Technik namens Namedropping verwendet, basiert auf dem *Name Server Transport Protocol* (NSTX) und setzt einen NSTX-konformen DNS-Server und -Client voraus. Der Server muss zusätzlich noch authoritativ zuständig für eine DNS-Domain sein (siehe Kasten *Im Internet*). Stellen wir uns vor, ein Hacker ist beispielsweise authoritativ für die Domain *baddomain.com* und kontrolliert ein System im durch eine Firewall geschützten Firmennetzwerk. Der Hacker möchte das System aus dem Internet fernsteuern und die Ausführungsergebnisse zugesendet bekommen.

Um die Daten vom kompromittierten System zum Hacker zu transportieren, stellt der DNS-Client einen DNS-Request nach einem speziell zusammengestellten Hostnamen wie zum Beispiel *b2xpdmVyIGthcm93.baddomain.com*, wobei es sich bei *b2xpdmVyIGthcm93* um die kodierte Nachricht handelt. Da der interne Nameserver für die Domäne *baddomain.com* nicht zuständig ist, leitet er die Anfrage an den zuständigen DNS-Server weiter. Da sich dieser unter der Kontrolle des Hackers befindet braucht er nur den angefragten Hostnamen zu dekodieren, und erhält so die gewünschten Daten von seinem kompromittierten System.

Damit besteht schonmal die Möglichkeit der Kommunikation in eine Richtung. Da aber der Hacker dem internen System neue Befehle zukommen lassen möchte, benötigt er einen weiteren Mechanismus. Hierzu schreibt der Hacker seine Befehle in einen TXT-Resource-Record. Hierbei handelt es sich um einen Freitext DNS-Record der im Normalfall unterschiedliche Einsatzgebiete wie zum Beispiel die Bereitstellung von PGP-Publiy-Keys hat. Sofern der Hacker auch hier wieder eine 7-Bit Kodierung verwendet, ist es für eine Firewall in der Regel nicht möglich zwischen einem gültigen TXT-Record und einer kodierten Nachricht zu unterscheiden.

Für weiterführende Informationen über Tunnelingangriffe wird empfohlen den Artikel *Firewall Piercing* zu lesen (siehe Kasten *Im Internet*).

Im Internet

- *http://cert.uni-stuttgart.de/archive/bugtraq/2001/04/msg00121.html* – Thomas Lopatic, *A fragmentation attack against IP Filter*,
- *http://www.ccc.de/congress/2004/fahrplan/files/221-firewallpiercing_21c3.pdf* – Maik Hensche & Frank Becker – *Firewall Piercing – Creative exploitation of valid Internet protocols*,
- http://www.thc.org/download.php?t=r&f=rwwwshell-2.0.pl.gz – eine HTTP-basierte Tunnelimplementation, rwwwshell,
- *http://www.csnc.ch/static/services/research/dnstunnel.html* – eine DNS-basierte Tunnelimplementation.

Tabelle 5. *Beispiele von Verwundbarkeiten in Firewallprodukten*

Produkt	Verwundbarkeit
Checkpoint Secure Platform	Firewall Rules Bypass Verwundbarkeit
Checkpoint VPN-1	ASN.1 Buffer Overflow
Checkpoint VPN-1	ISAKMP Buffer Overflow
Cisco IOS Firewall	Authentication Proxy Buffer Overflow
Cisco Catalyst 6500/6700	FW Services Module ACL Bypass Verwundbarkeit

Firewall Verwundbarkeiten

Die Sicherheit eines Netzwerkes steht und fällt mit der eigenen Sicherheit der Firewall. Sofern eine Firewall selbst verwundbar für Angriffe wie beispielsweise Buffer-Overflows ist, stellt sie für einen Angreifer kein Hindernis mehr dar, da dieser die Firewall einfach nach seinen Bedürfnissen umkonfigurieren kann. Auch kann der Angreifer die Firewall selbst dazu verwenden, Angriffe in das interne Netzwerk zu starten, was nur durch eine mehrstufige Firewallumgebung zu verhindern ist.

Remote ausnutzbare Verwundbarkeiten in namhaften Firewallprodukten werden leider sehr oft entdeckt. Um einen Überlick über aktuell existierende Verwundbarkeiten zu bekommen, empfiehlt es sich die Vulnerability-Datenbank auf *www.securityfocus.com* anzuschauen. Einige Beispiele von Verwundbarkeiten der letzen Monate sind in Tabelle 5 aufgeführt.

Zusammenfassung

Es existieren etliche Möglichkeiten eine Firewall zu umgehen. Einige von ihnen basieren auf der unzureichenden Funktionalität des Produktes, und andere auf einer mangelhaften Konfiguration oder gar auf Schwachstellen in der Firewallsoftware selbst. Dennoch kann durch die Implementierung einer mehrstufigen Firewallumgebung unter Einsatz aktueller Firewalltechnologien, in Verbindung mit regelmäßigen Sicherheitsaudits ein guter Schutz für interne Systeme erzielt werden. ●