– RDUe Labor –
Rechnernetze und Datenübertragung
Laborbericht über Versuch:
Firewalling und Attacken

Andreas Hofmeier

Auftraggeber:	Prof. Dr.-Ing. Kai-Oliver Detken, Fachhochschule Bremen
Durchführung am:	04.06.2004, 08.06.2004 und 16.06.2004
Ort der Durchführung:	FH Bremen, Flughafenalle 10, Raum: I222
Abgabe am:	28.07.2004

Versuchsgruppe

Andreas Hofmeier
Clemens Nau
Marco Bensch

Zusammenfassung

.
In diesem Versuch sollte die Leistung und die Sicherheit einer unter SuSE GNU Linux laufenden Firewall untersucht werden. Mit Hilfe eines speziellen Computers (Navtel InterEmulator) wurden der HTTP-Server, die HTTP-Klienten sowie die Angriffe Simuliert.

Für den Versuch vorgesehen waren zusätzlich Versuche mit einem unter Linux laufenden HTTP-Server und einer FTP-Übertragung vom InterEmulator zum InterEmulator. Diese Versuche konnten aufgrund technischer Probleme nicht durchgeführt werden.

Der Ping (DoS) Angriff legte die Firewall wie erwartet lahm. Erwartungsgemäß reagierte die Firewall garnicht auf Ping-of-Death-Attacken. Die Versuche mit der Smurf sowie der SYN-Attacke wurden nicht korrekt durchgeführt.

Inhaltsverzeichnis

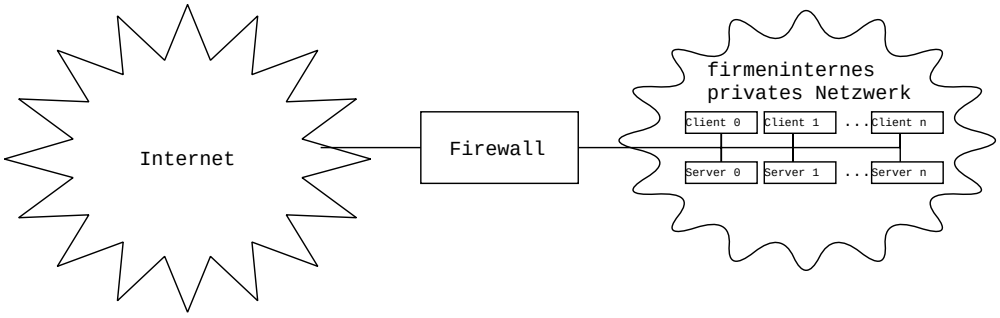
1 Versuchsziele

Ziele dieses Versuches war es ein Grundverständnis für Netzwerksicherheit zu erlangen. Es sollte ausprobiert werden, welche Formen von Attacken eine Firewall abwehren kann und welche nicht. Zusätzlich sollte die Performance der Firewall bestimmt werden, denn keine Sicherheit ohne Preis (hier die Zugriffszeit).

2 Vorbetrachtungen

2.1 Firewall - Was ist das?

Bei einer Firewall handelt es sich um einen Computer welcher ein sicheres Netzwerk (z.B. ein firmeninternes Netzwerk) von einem unsicheren Netzwerk (z.B. dem Internet) abschottet.



(Im Internet gibt es natürlich unzählige Klienten und Server, dass ich habe hier vorausgesetzt. Größenverhältnisse nicht ganz realistisch...)

Wollte man hundertprozentige Sicherheit erreichen, so müsste man “nur” die Verbindung der beiden Netzwerke trennen. Hat dann allerdings auch nur 0% Zugriff auf Dienste außerhalb des internen Netzwerkes. Möchte man unbeschränkten Zugriff auf diese externen Dienste haben, ist es ein leichtes die beiden Netzwerke über zum Beispiel einen Router miteinander zu verbinden. Nun hat man allerdings keinen Schutz mehr vor der “bösen Welt”. Schnell wird klar, dass keines der beiden extreme wirklich sinnvoll ist. Aber es sollte genauso klar sein, dass alles seinen Preis hat. Es ist nahezu unmöglich hundertprozentige Sicherheit und vollen Zugriff unter ein Dach zu bringen.

Möchten Sie zum Beispiel ein Firmengelände einer sicherheitsbewussten Firma betreten, so werden sie um Eingangskontrollen kaum herumkommen. Die Kontrollen sind der Preis für die Sicherheit. Nicht nur das sie eine gewisse Zeit warten müssen, auch das Sicherheitspersonal und die Ausweise, etc kosten Geld. Genauso ist es bei einer Firewall. Eine Firewall ist sozusagen die Eingangskontrollstelle zu ihrem Netzwerk. Auch wenn die eine Firewall ein Paket meist

schneller abfertigt als ein Wachmann eine Person, kann die Verzögerung schon zu Problemen führen. Ein gutes Beispiel hierfür sind Echtzeitanwendungen wie Videokonferenzen.

2.2 Angriffe auf Firewalls I: Denial-of-Service (DoS) Attacke

Ein weiteres Problem einer Eingangskontrolle ist deren Dimensionierung. Aus Kostengründen wird man natürlich versuchen eine auf seinen Bedarf zugeschnittene Eingangskontrolle aufzubauen. Aber was passiert, wenn sich alle Bewohner einer Stadt entschließen zu der Eingangskontrolle zu gehen? Gut, sie würden nicht reingelassen – aber, da die Zufahrtsstraßen verstopft wären, würde auch niemand anders mehr reinkommen. Da Computer "normalerweise" einfacher zu programmieren sind als Menschen, ist es in Netzwerken relativ einfach genügend viele Zugriffe zu produzieren um die normale Funktion eines System untergehen zu lassen. So etwas würden man dann eine Denial-of-Service Attacke nennen.

2.3 Paketfilter

Wie der Name vermuten lässt, filtert diese einfache Form der Firewall Pakete nach gewissen Kriterien. So können zum Beispiel alle Pakete ausgefiltert (nicht zugelassen) werden, die zu groß sind, einen ungültigen oder unerwünschten Absender haben oder an einem bestimmten Port/Service (der nicht erwünscht ist) gerichtet sind.

Paketfilter eignen sich gut um zum Beispiel eine bestimmten Service zu unterbinden sind aber nutzlos gegen Angriffe auf Applikationsebene. So könnten unerwünschte FTP-Zugriffe durch sperren des Ports 21 unterbunden werden. Dies ist zum Beispiel sinnvoll, wenn der FTP-Server nur intern verwendet werden soll. FTP-Anfragen werden schon vor erreichen des Servers abgefangen und können diesen nicht mehr beeinträchtigen. Würden die Anfragen bis zum FTP-Server durchgelassen, könnte dieser zum Beispiel durch Überflutung mit Zugriffen (Denial-of-Service Attacke) beeinträchtigt werden. Des weiteren wäre es möglich Schwachstellen des Servers selber auszunutzen. Hier sind insbesondere Buffer-Overflows zu nennen. Vor diesen beiden Angriffen bietet ein Paketfilter also nur dann Schutz, wenn der entsprechende Service verboten wurde. Dies auch nur begrenzt, da die Firewall selber so damit beschäftigt sein kann die Angriffe abzuweisen, dass nichts anderes mehr durchkommt. Einige Beispiele: Ping-Flooding (DoS auf Paketebene) gegen FTP-Server (über Firewall; egal, ob FTP zugelassen): Die die Internetverbindung wäre lahmgelegt aber der FTP-Server könnte intern weiter verwendet werden; Überflutung mit FTP-Zugriffen: Ist der Service erlaubt, also von außen sichtbar, ist eine Firewall machtlos. Ansonsten würde die Firewall den Angriff total abwehren. Die Internetverbindung wäre nur wenig darurch belastet. Weiterhin kann ein Paketfilter manipulierte

Paket filtern. So zum Beispiel Ping-of-Death-Attacken, welche mit Hilfe von manipulierten (z.B. übergroßen) Paketen versuchen den Zielrechner zum Absturz zu bringen. Das meint aber nicht, dass die Firewall in der Lage wäre manipulierte Daten innerhalb eines korrekten Datenstroms (z.B. Ausnutzungen eines Buffer-Overflows) erkennen und abzuweisen. Hierfür ist der jeweilige Server selber zuständig. Um diesem Problemen vorzubeugen können Applikationsfilter eingesetzt werden.

2.4 Applikationsfilter

Bei Applikationsfiltern handelt es sich um einen Filter auf Applikationsebene. Ganz im gegensatz zu Paketfiltern können hier Angriffe auf der Applikationsebene (z.B. Ausnutzung von Buffer-Overflows) abgewehrt werden. Applikationsfilter müssen daher auf die jeweilige Applikation zugeschnitten sein, anders als Paketfilter, denen die Applikation egal ist. Proxys sind zum Beispiel eine Form von Applikationsfiltern.

Ein weiteren interessanter Aspekt neben dem Filtern von sicherheitsrelevanten Daten wie Viren und Trojanern, ist das Filtern von unerwünschten Inhalten, wie zum Beispiel Werbung, Spam, Pornos, etc...

2.5 Proxys

Proxys sind eine Erweiterung von Applikationsfiltern. Sie filtern nicht nur den Datenstrom sondern speichern ihn auch zwischen (cachen). Das hat den großen Vorteil, dass häufig gebrauchte Daten nicht immer wieder durch einen Flaschenhals (z.B. langsame Internetverbindung) neu geladen werden müssen. Das macht den Zugriff schneller. In der heutigen Zeit, in der nach Traffic-Volumen abgerechnet wird, macht es sich ebenfalls in den Onlinekosten bemerkbar. Da häufig gebrauchte Daten eben nur einmal, statt “häufig” geladen werden.

Für die häufigen HTTP (WWW/WEB) Zugriffe stehen unter GNU Linux unter anderem diese beiden Proxys zur Verfügung: Apache (kann neben WEB-Server auch als Proxy arbeiten) und Squid.

2.6 Angriffe auf Firewalls II

Angreifen kann man nur, was man auch erkennen kann. Also sollte die Firewall von außen nicht sichtbar sein. Dies kann dadurch erreicht werden, dass die Firewall nicht auf ICMP-Anfragen (z.B. Pings) reagiert. So bleibt die Firewall für Traceroute unsichtbar.

Das setzt natürlich voraus, dass die IP der Firewall für nichts anderes als für Firewall verwendet wird. Läuft zum Beispiel der WEB-Server auf dem selben Rechner wie die Firewall, ist die IP der Firewall schon bekannt und angreifbar. Ganz davon zu schweigen, dass der zusätzliche Dienst ebenfalls Sicherheitslücken enthalten kann und somit die Firewall "angreifbarer" macht. Hier gilt: Jeder Service erhöht die Angriffsfläche.

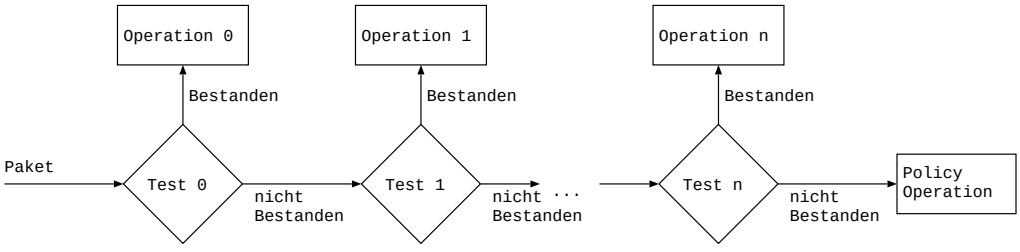
Gelingt ein Einbruch auf einer Firewall (zum Beispiel über ein WEB-Server auf dem selben Rechner), kann diese zum Router umfunktioniert werden und alle Zugriffe von außen sind möglich. Bricht jemand allerdings auf einem "nur Webserver"-Rechner ein, so sind die Zugriffe von außen werterhin durch die Firewall beschränkt. Ein von einem Trojaner geöffneter Port kann zum Beispiel nicht von außen erreicht werden.

Natürlich sollte der Admin bei einem Login auf einer Firewall (genauso wie bei jedem anderen Login auch) alle Sicherheitsvorschriften beachten. Einloggen wenn möglich nur von "sicheren Rechnern" (keine Fremdrechner mit Keyboardloggern im Hintergrund) aus. Ebenfalls sollte ein anerkanntes Verschlüsselungsverfahren wie z.B. SSL/SSH verwendet werden.

2.7 Paketfilter unter Linux

Der Linux-Kernel hat einen Paketfilter integriert. Auf Grund der hohen Zuverlässigkeit und Sicherheit eignet sich GNU Linux ganz besonders gut für den Einsatz als Firewall.

2.7.1 Bedingungsketten



Bei jedem Paket geht der Linux-Kernel die entsprechende Bedingungskette durch. Eine Bedingungskette besteht aus einer Reihe von Tests und bedingten Operationen die nacheinander abgearbeitet werden. Jede Kette endet mit einer Policy. Sozusagen einer "wenn nichts anderes zutrifft, dann tue dies"-Operation. Unter Tests können zum Beispiel Paketgröße, Quell- und Zieladresse sowie Port, Pakettyp (UDP/TCP/ICMP), Zuordnung (bereits offene Verbindung, Verbindungsanfrage), etc getestet werden. Die Operation ist jeweils einem Test zugeordnet und wird durchgeführt wenn dieser zutrifft. Operationen können sein: Pa-

ket durchlassen (akzeptieren, keine weiteren Tests), Paket Verwerfen (Löschen, keine weiteren Tests) oder Protokollieren (Eintrag an Systemlogdeamon/Logfile senden, die Kette wird weiter durchlaufen)

Um die Tests und die daraus resultierenden Operationen so exakt wie möglich festlegen zu können, existieren mehrere Ketten die bei verschiedenen Ereignissen durchlaufen werden: INPUT (wird bei jedem an dem Firewallrechner eintreffenden Paket abgearbeitet), OUTPUT (wird bei jedem Paket durchlaufen, welches den Firewallrechner verlässt) und FORWARD (mit welcher jedes Paket überprüft wird, welches von der Firewall weitergegeben, bzw geroutet wird). Zusätzlich zu diesen Ketten gibt es noch weitere Ketten die zum Beispiel bei einem durch NAT bearbeiteten Paket zum Einsatz kommen. Des weiteren kann der Benutzer auch eigene Ketten definieren, die er mittels einer Operation aufrufen kann.

Es ist zu beachten, dass die Ketten "von links nach rechts bzw von oben nach unten" abgearbeitet werden. Wenn also die erste Anweisung ein ACCEPT ALL (also: Test immer Richtig und Operation ACCEPT) ist, werden alle Pakete angenommen, egal, was für Anweisungen noch folgen. Ein Sicherheitsbewusster Admin definiert grundsätzlich alle ACCAPTS und schmeißt mit Hilfe der Policy alles was übrigbleibt weg. Auf keinen Fall sollte es umgekehrt gehandhabt werden, weil in diesem Fall etwas Sicherheitsrelevantes übersehen werden könnte. Wird im anderen Fall etwas übersehen, zeigt sich dies daran, dass ein bestimmte Zugriff nicht mehr funktioniert (z.B. HTTP). Dies ist einfach zu erkennen und schnell zu beheben. Jedenfalls einfacher als die Tatsache, dass ein Zugriff durchgelassen wird, der nicht durchgelassen werden sollte. Wenn dieser Zugriff vergessen wurde, ist davon auszugehen, dass dieser auch beim obligatorischen Test der Firewall vergessen wird.

2.7.2 iptables: Setzen der Bedingungsketten

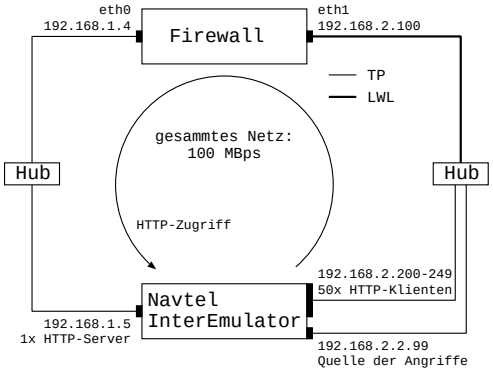
Die Ketten liegen im Hauptspeicher und können mittels dem Programm `iptables` ausgelesen und geändert werden. Es handelt sich hierbei um ein kleines Konsole-Programm, welches vorwiegend in Skripten Verwendung findet oder von höheren, meist graphischen, Konfigurationsprogrammen aufgerufen wird. Es ist zu bedenken, dass der Hauptspeicher flüchtig ist. Alle Ketten also verloren gehen, wenn der Computer ausgeschaltet wird. Um die Ketten auf Festplatte zu sichern existieren vorgefertigte Skripte.

3 Versuchsaufbau und Durchführung

3.1 Das Netzwerk

Der Versuch wurde im großen und ganzen so wie in der Anleitung beschrieben aufgebaut: Der InterEmulator-Client wurde über das Netzwerk 192.168.2.0/255.255.255.0 mit der Firewall verbunden. Die Firewall reichte die Anfragen dann über das Netzwerk 192.168.1.0/255.255.255.0 an den InterEmulator-Server weiter.

Die Beiden Netzwerke waren völlig unabhängig zu einander. Bei beiden Netzwerken sowie allen anderen beteiligten Netzwerkequipment handelte es sich um 100MBps Geräte.



Im ersten Schritt des Versuches, nachdem die Hubs aufgestellt und entsprechend verkabelt waren, folgte die Konfiguration der einzelnen Rechner. Dies erfolgte in Gruppenarbeit weitgehend parallel. Es ist zu erwähnen, dass es nicht so einfach war, den Versuch zum laufen zu bekommen, als dies erwartet worden war.

3.2 Firewall

Konfiguration der GNU Linux Firewall:

3.2.1 Netzwerkkonfiguration

Einstellung	eth0 (TP)	eth1 (LWL)
MAC-Adresse	00:30:84:6C:3A:BE	00:04:76:DA:F6:BA
IP-Adresse	192.168.1.4	192.168.2.100
Netmask	255.255.255.0	255.255.255.0
Default Gateway	–	–

Das Kernel-Modul für den Paketfilter war bereits ordnungsgemäß geladen.

Ein Problem, welches uns einige Zeit gekostet hat, bestand darin, dass wir erst rausfinden mussten, wieso die Firewall keine Pakete weiterleitet. Des Rätsels Lösung war, dass das Forwarding erst eingeschaltet werden musste:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

3.2.2 Konfiguration des Paketfilter über iptables

Die Konfiguration des Paketfilter erfolgte über iptables:

```
# Policy fuer alle drei verwendeten Ketten auf Paket verwerfen
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
# Alle zu einer bereits bestehenden Verbindung (Stream)
# gehoerenden Pakete zulassen
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
# Vom Firewallrechner gesendete ICMP-Pakete zulassen,
# damit das Anpingen und testen der Server und Klienten
# moeglich ist
iptables -A OUTPUT -p icmp -j ACCEPT
# Das Aufbauen einer HTTP-Verbindung im Protokoll vermerken
iptables -A FORWARD -p tcp --dport 80 -m state --state NEW -j LOG \
  --log-prefix "New HTTP Connection "
# Das Aufbauen einer HTTP-Verbindung erlauben
iptables -A FORWARD -p tcp --dport 80 -m state --state NEW -j ACCEPT
# Das Aufbauen einer FTP-Verbindung im Protokoll vermerken
iptables -A FORWARD -p tcp --dport 21 -m state --state NEW -j LOG \
  --log-prefix "New FTP Connection "
# Das Aufbauen einer FTP-Verbindung erlauben
iptables -A FORWARD -p tcp --dport 21 -m state --state NEW -j ACCEPT
# Alle weiter zu leitenden DNS-Anfragen verwerfen
iptables -A FORWARD -p udp --dport 53 -j DROP
# letzte Anweisung, bevor uebrig gebliebene Pakete durch die
# Policy verworfen werden: Dies im Protokoll vermerken
iptables -A FORWARD -j LOG --log-prefix "DEFAULT DROP FORWARD "
iptables -A INPUT -j LOG --log-prefix "DEFAULT DROP INPUT "
iptables -A OUTPUT -j LOG --log-prefix "DEFAULT DROP OUTPUT "
```

3.3 Konfiguration des Navtel InterEmulator

Der folgende Abschnitt beschreibt die Konfiguration des Navtel InterEmulators.

Der Navtel InterEmulator ist ein PC in einem Spezialgehäuse. Durch spezielle Karten ist es dem Gerät möglich große Netzlasten in Form von Zugriffen oder Angriffen zu erzeugen.

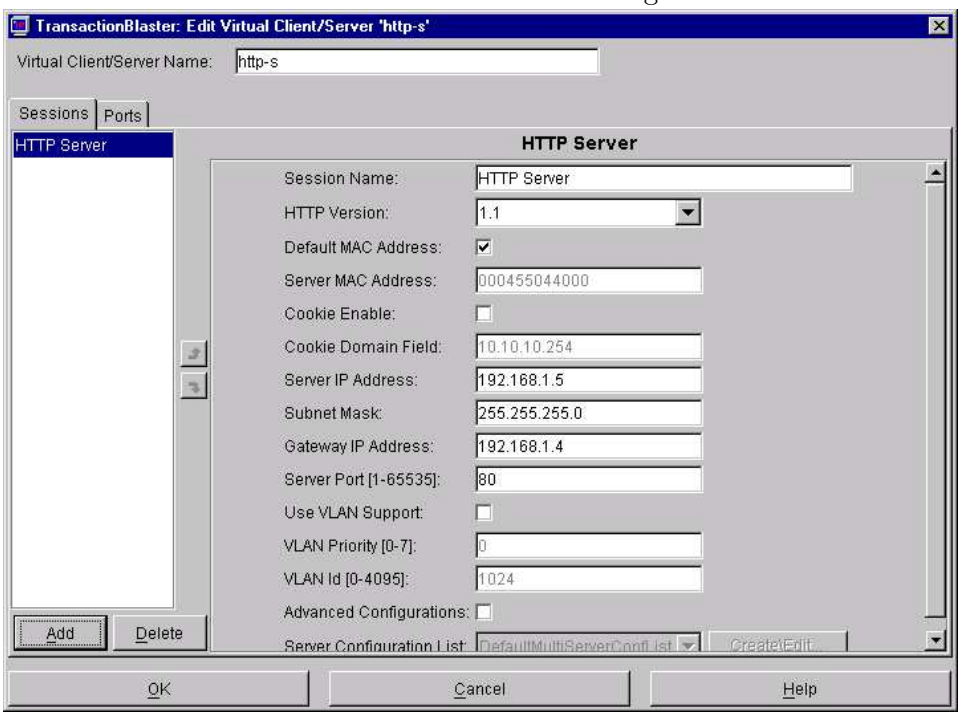
Es ist an dieser Stelle zu erwähnen, dass wir bei der Konfiguration auf erhebliche Probleme stießen. Der Versuch, welcher für eine Labor-Sitzung angesetzt war, musste auf 3 Laborsitzungen zu je meist mehr als 5 Stunden ausgedehnt werden. Der InterEmulator Reagierte teilweise unlogisch und nicht nachvollziehbar auf gewisse Einstellungen.

So mussten wir zum Beispiel feststellen, dass die Ports (Netzwerkanschlüsse) des InterEmulators nicht beliebig beschaltet (mit Servern, Klienten oder Angriffen belegt) werden konnten. Die Anordnung musste zu Teil ausprobiert werden.

Versuche den HTTP-Versuch über den InterEmulator zu fahren funktionierten einmal und ein anderes mal wieder nicht – ohne erkennbaren Unterschied in der Konfiguration. Erschwerend kam hinzu, dass keine ausreichende Dokumentation zur Verfügung stand.

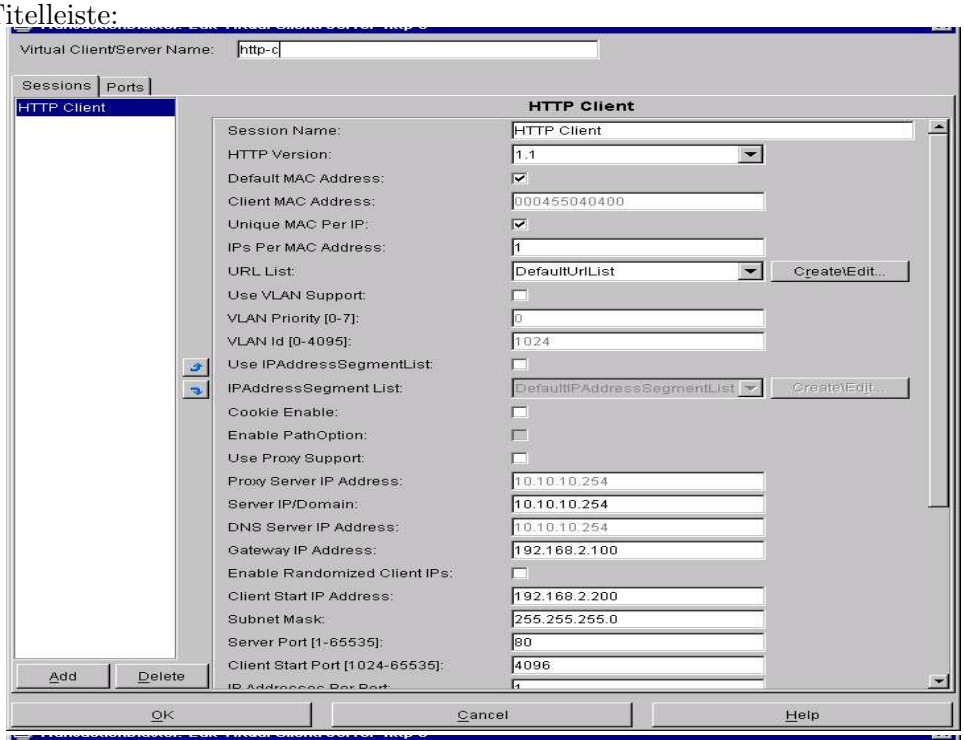
3.3.1 HTTP-Server

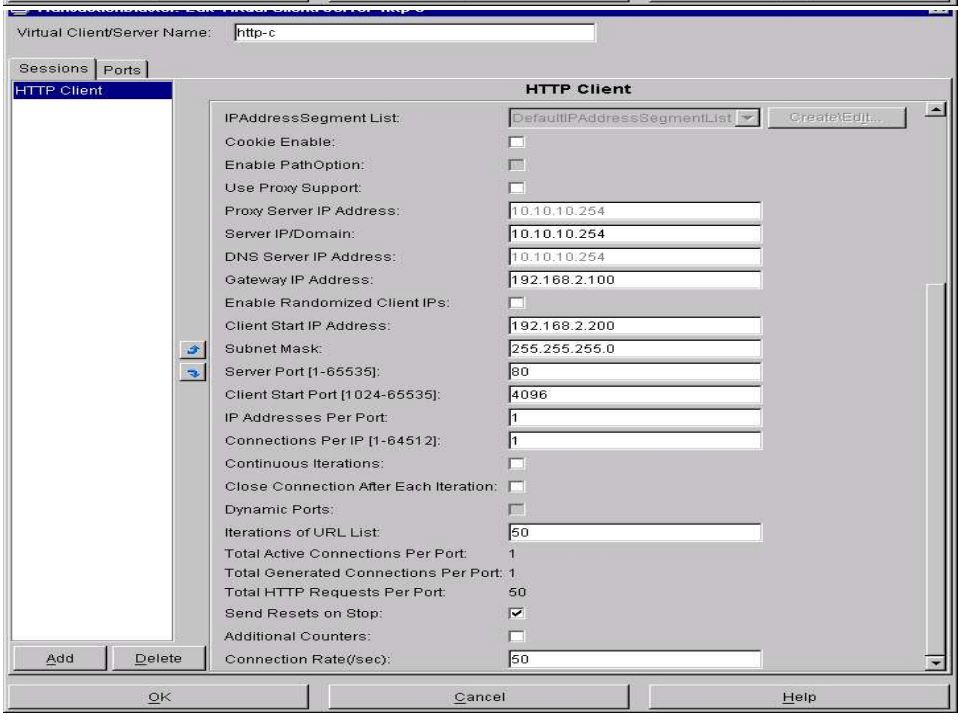
Der HTTP-Server des InterEmulators wurde mit folgenden Werten betrieben:



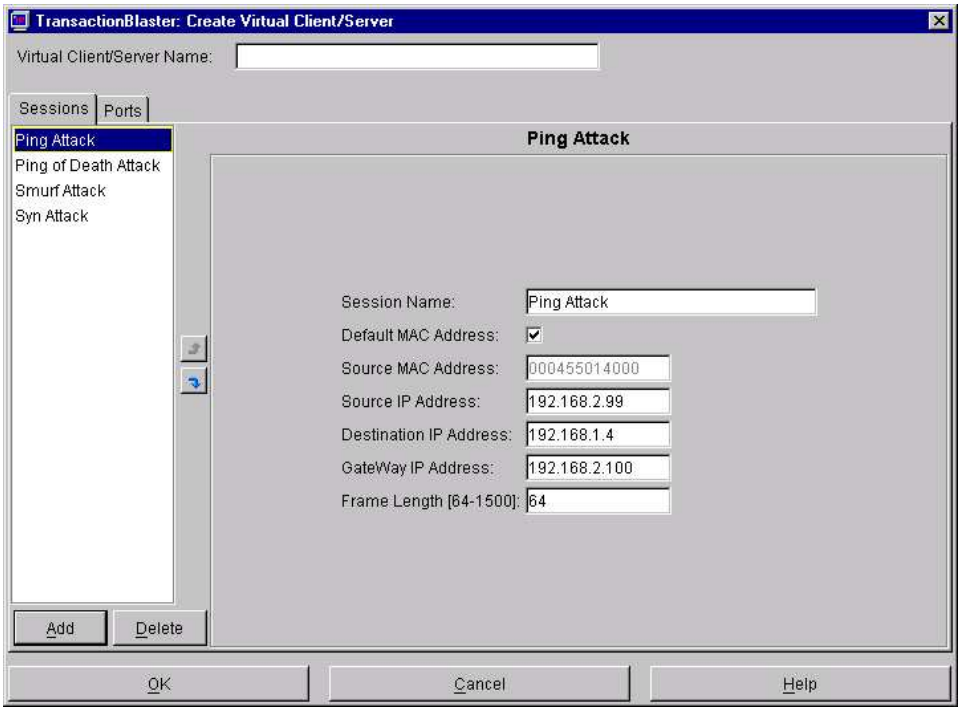
3.3.2 Konfiguration der HTTP-Klienten

Es wurden 50 HTTP-Klienten durch den InterEmulators emuliert. Da das Fenster mit den Einstellungen nicht auf eine Bildschirmseite passte, musste es in zwei Teilen dargestellt werden. Das ist auch der Grund für die abgeschnittene Titelleiste:

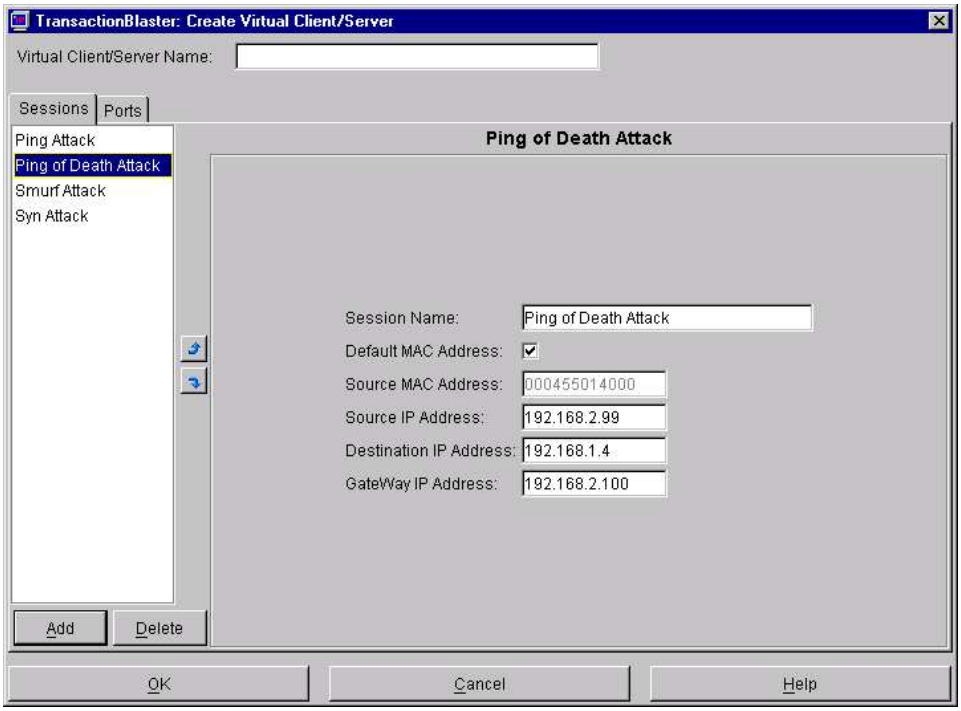




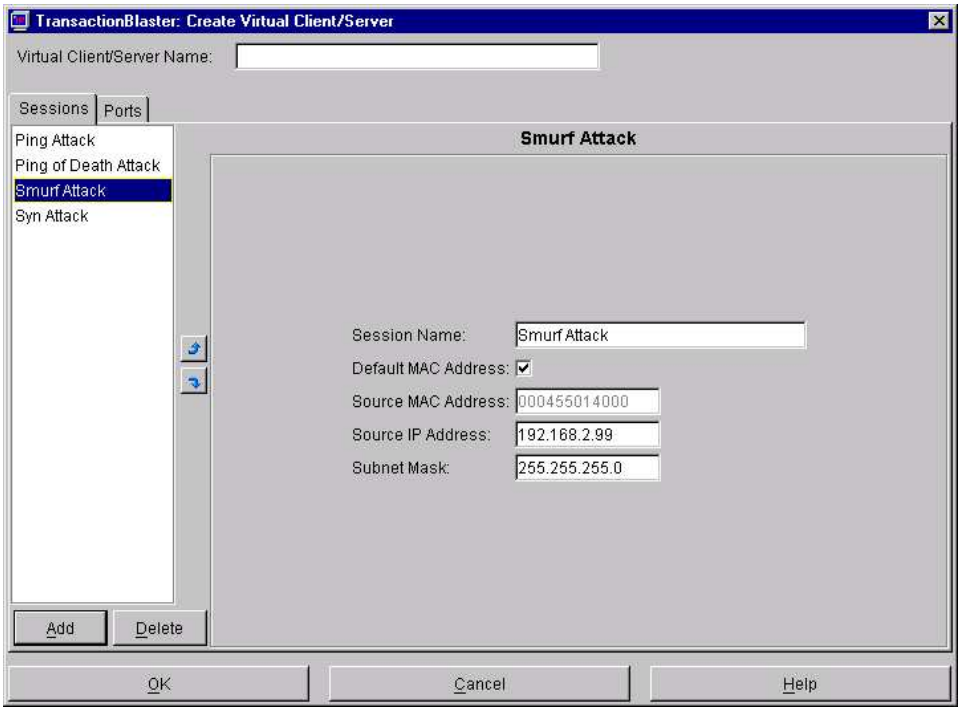
3.3.3 Konfiguration der Ping Attacken



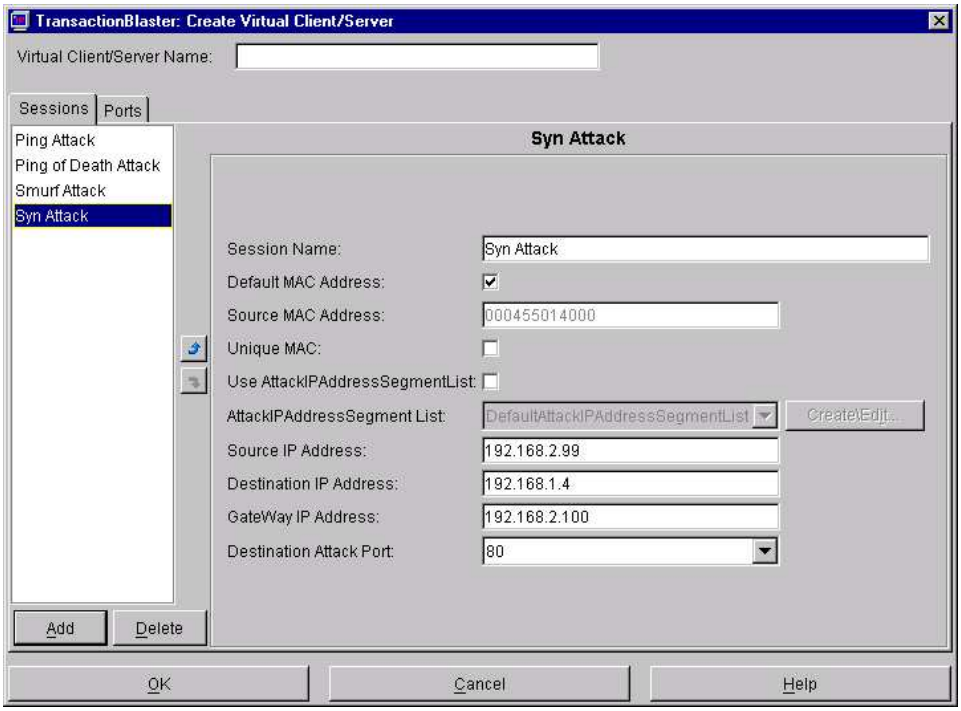
3.3.4 Konfiguration der Ping-of-Death Attacken



3.3.5 Konfiguration der Smurf Attacken



3.3.6 Konfiguration der SYN Attacken



4 Versuchsauswertung

4.1 Übersicht Transfer zu Attacken

Die folgende Tabelle zweigt die vom Navtel InterEmulator "abgelesenen" Werte. Es werden hier die gesendeten und empfangenden Frames gegenübergestellt.

		Client			Server	
Attack	FW	Rx	Tx	Time	Rx	Tx
Ping	1758	6938	8940	77	8597	6838
Ping-of-Death	3203	13277	15950	67	15804	12604
Smurf	4161	311001	16500	69	16303	13003
SYN (Ref)	3602	14404	18000	74	17806	14205

Die Angaben zur Laufzeit wurden an Hand des Protokollfiles der Firewall überprüft. Die vom InterEmulator gemessenen Laufzeiten waren immer 4 Sekunden größer. Alle Laufzeitangaben in Sekunden.

Der Server wurde immer zuerst gestartet und zuletzt beendet. Daten wurden nur übertragen, wenn Server und Klient gleichzeitig liefen. Daher ist die Zeit, die der Server lief, nicht von Interesse. Die Counter wurden für jeden Versuch durch einen Neustart des Server, Klienten oder Attacke auf Null zurückgesetzt.

Tx und Rx wurden bei dem Server und Klienten jeweils in "Frames" gemessen. Leider ist nicht genau bekannt, was ein Frame ist. Es ist anzunehmen, dass es sich hierbei um ein Ethernet-Frame handelt. Also ein Paket darstellt. Die Größe ist leider nicht bekannt.

Die unter FW angegebenen Werte stellen die Anzahl der Erstellungen von Verbindungen dar, die vom Paketfilter der Firewall festgestellt wurden. Also, wann hat ein Client eine Socket-Stream-Verbindung zu einem Server aufgebaut. Dieser Wert entspricht der Anzahl der "New HTTP Connection" - Zeilen im Protokollfile.

Da nicht bekannt ist, wieviele Frames pro Stream übertragen wurden, oder ob es da überhaupt einen Zusammenhang gab, sind dieser Werte leider nicht mit den Rx und Tx-Werten vergleichbar. An dieser Stelle hätten wohl die übertragenen Pakete statt die Streams gezählt werden sollen. Dazu hätte die iptable anders Eingestellt werden müssen.

Um eine bessere Übersicht zu erhalten wurden diese Zahlen auf die Laufzeit des jeweiligen Versuches bezogen. Also der jeweilige Wert durch die entsprechende Laufzeit dividiert.

		Client		Server	
Attack	FW/s	Rx/s	Tx/s	Rx/s	Tx/s
Ping	22.8	90.1	116.1	111.6	88.8
Ping-of-Death	47.8	198.2	238.1	235.9	188.1
Smurf	60.3	4507.3	239.1	236.3	188.4
SYN (Ref)	48.7	194.6	243.2	240.6	192.0

Da bei der SYN-Attacke keine Attacken gesendet wurden, kann diese Zeile mit Werten hier als Referenz verwendet werden. Die vorgesehene Referenzmessung wurde zwar durchgeführt, es wurde aber versäumt, die Werte zu notieren.

Es ist zu erkennen, dass die Ping-Attacke den Datentransfer am meisten heruntergezogen hat. Während die Ping-Attacke lief, kamen am wenigsten Pakete durch.

Die Ping-of-Death Attacken haben den Transfer nicht sonderlich beeinflusst. Dies kann darauf zurückgeführt werden, dass nur wenige Attacken gesendet wurden, welche von der Firewall abgefangen wurden.

Sieht man von dem Ausreißer bei Rx/Client mal ab (dazu gleich mehr), so hat der Smurf-Attacke den Datentransfer noch etwas weniger beeinflusst als die Ping-of-Death Attacke.

4.2 Übersicht Transfer Client/Server

Diese Tabelle stellt eine Übersicht über das Verhältnis von gesendeten und Empfangenen Frames dar. Es wurden jeweils die Anzahl der Empfangenden Pakete durch die Anzahl der Gesendeten geteilt. Dieser Wert wurde dann mit 100 multipliziert um auf Prozent zu kommen.

	Daten (Frames) Angekommen (in Prozent)	
Attack	Server → Client	Client → Server
Ping	101.5	96.2
Ping-of-Death	105.3	99.1
Smurf	2391.8	98.8
SYN	101.4	98.9

Dieser Gegenüberstellung Zeigt, dass fast alle oder sogar mehr der gesendeten Frames angekommen sind. Dies ist meiner Ansicht nach nicht möglich. Während der Ping-Attacke zum Beispiel, war die Firewall tot. Das Netz war überlastet. Wie können in diese Fall 96% der Pakete durchkommen? Geschweige denn 101%? Da die Attacken fast 50% der Zeit liefen, erwarte ich, dass um die 60% der Pakete durchkommen.

Das bei der Smurf-Attacke ca. 24 mal so viele Frames angekommen sind, wie

abgeschickt wurden, kann damit erklärt werden, dass die Smurf-Attacken die Firewall passiert haben und vom InterEmulator als Zugriffe gewertet wurden. Diese These wird auch von der Tatsachen unterstützt, dass die Firewall erheblich mehr Streams registriert hat, als bei der Referenzübertragung.

4.3 Anzahl der Attacken

In dieser Tabelle ist die Anzahl der jeweils durchgeführten, empfangenen (Werte vom InterEmulator) und von der Firewall registrierten Attacken dargestellt.

Die Anzahl der von der Firewall registrierten Angriffe wurde durch Auszählung der entsprechenden Zeilen im Protokollfile ermittelt. Dazu im Abschnitt Wirkung der Angriffe auf die Firewall mehr.

	Attack			
Attack	FW	Rx	Tx	Time
Ping	668	1	4671789	32
Ping-of-Death	118	1	9070	21
Smurf	279323	16974	3111350	20
SYN	0	0	0	29

Übersicht Angriffe pro Sekunde.

	Attack		
Attack	FW/s	Rx/s	Tx/s
Ping	20.9	0.0	145993.4
Ping-of-Death	5.6	0.0	431.9
Smurf	13966.2	848.7	155567.5
SYN	0.0	0.0	0.0

4.4 Wirkung der HTTP-Zugriffe auf die Firewall

Die Firewall registrierte, wie mittels iptables eingestellt, nur das Aufbauen einer neuen Socket-Stream-Verbindung. Jede neue Verbindung, bzw. jedes Paket, welches eine Verbindung aufbaut, wurde in der Logdatei mittels einer Zeile festgehalten. Hier ein Beispiel:

```
Jun 16 15:28:55 INRN05 kernel: New HTTP Connection IN=eth1 \
OUT=eth0 SRC=192.168.2.200 DST=192.168.1.5 LEN=44 TOS=0x00 \
PREC=0x00 TTL=59 ID=0 DF PROTO=TCP SPT=4096 DPT=80 WINDOW=8\
192 RES=0x00 SYN URGP=0
```

(Die Zeile aus der Logdatei war zu lang um auf eine Seite zu passen, daher wurde sie umgebrochen, was durch einen Backslash am Ende der Zeile gekennzeichnet

wird.)

Der erste Teil dieser Zeile gibt das Datum, die Uhrzeit, den Namen des Computers und das verursachende Programm (hier der Paketfilter, welcher in den Kernel eingebettet ist) an: `Jun 16 15:28:55 INRN05 kernel`

Nun folgt das "log-prefix": `New HTTP Connection`. Das Interface, an welchem dass Paket ankommt bzw die Firewall wieder verlässt sowie die Quell- und Zielrechner dürfen natürlich auch nicht fehlen: `IN=eth1 OUT=eth0 SRC=192.168.2.\ 200 DST=192.168.1.5`

Hiernach folgen noch einige Paketinformationen, wie die Time-To-Live, Paketgröße, Protokoll, Windowgröße oder Typ.

Das jede Verbindung eine eigene Zeile hervorgerufen hat und nicht mit `last message repeated n times` abgekürzt wurde, ist darauf zurück zu führen, dass der InterEmulator für die nächste Verbindung die nächste IP verwendet hat. Also 192.168.2.200, 192.168.2.201, ... 192.168.2.249 und dann wieder von vorne. Alle aufeinanderfolgende Zeilen also unterschiedlich waren.

Pro Sekunde wurden 50 Verbindungen aufgebaut, woraus man schließen kann, dass jeder der 50 Klienten einmal pro Sekunde seine Verbindung neu aufgebaut hat.

4.5 Wirkung der Angriffe auf die Firewall

4.5.1 Ping Attack

Während der Ping Attacken liefen, reagierte der GNU Linux Rechner, welcher als Firewall herhalten musste, überhaupt nicht. Es kam also zu dem erwarteten Ergebnis, das die Firewall zwar die Pings abschottet, währenddessen aber auch nichts anderes durchkommt. Der Computer reagierte noch nicht einmal auf Eingaben über die Text-Konsole.

Das steht allerdings im Wiederspruch zu dem weiter oben abgeleiteten Ergebnis, dass die meisten Frames durchkamen. Ich hätte erwartet, dass viel mehr Frames gesendet wurden (in beiden Richtungen) als Empfangen wurden.

Dieser Wiederspruch kann aufgelöst werden, indem man davon ausgeht, dass nur Frames übertragen werden, wenn eine Socket-Stream-Verbindung besteht. Während der Angriffe wurden laut Protokollfile tatsächlich keine Verbindungen aufgebaut.

Erwähnenswert ist noch, dass es eine Lücke von 14 Sekunden im Protokollfile

zwischen der letzten neuen Verbindung und dem ersten eingetragendem Angriff gibt.

Einige der Zeilen im Protokollfile sind durcheinander geraten. Das meint, dass eine neue Zeile begonnen wurde, bevor die alte abgeschlossen war.

Dies alles sind Indizien für die Tatsache, dass der Kernel (bzw. der Paketfilter) völlig überlastet war und seinen Dienst nicht mehr richtig verrichtete.

Beispiel für eine durch den Angriff verursachten Zeile im Logfile. Da diese Zeilen alle gleich sind oder besser gesagt gleich sein sollten, wurde hier mittels `last message repeated 34 times` abgekürzt. Diese Form der Abkürzung wurde natürlich beim auszählen der Einträger für die weiter oben liegenden Tabellen berücksichtigt.

```
Jun 16 15:29:37 INRN05 kernel: DEFAULT DROP FORWARD IN=eth1\
 OUT=eth0 SRC=192.168.2.99 DST=192.168.1.5 LEN=46 TOS=0x00 \
PREC=0x00 TTL=127 ID=60929 PROTO=ICMP TYPE=8 CODE=0 ID=1 SE\
Q=1
Jun 16 15:29:37 INRN05 last message repeated 34 times
```

Da wir mittels iptables keine Regel definiert hatten, die Pakete wie dieses ausschließt, wurd es durch die DORP-Anweisung in der Policy gekillt. Den Eintrag haben wir mit Hilfe der letzten Anweisung in der FORWARD-Chain erzwungen. Dieser letzte Eintrag sorgt nur für eine Ausgabe im Logfile, er schmeißt das Paket nicht weg, dies ist Aufgabe der Policy.

Hier ein Beispiel für **eine** vermanschte Zeile im Logfile:

```
Jun 16 15:29:37 INRN05 kernel: DEFAULT DROP FORWARD IN=eT=e\
th0 SRC=192.168.2.99 DST=192.168.1.5 LEN=46 TOS=0x00 PREC=0\
x00 TTL=127T=eth0 SRC=192.168.2.99 DST=192.168.1.5 LEN=46 T\
OS=0x00 PREC=0x00 TTL=127 ID=60929 PT=eth0 SRC=192.168.2.99\
 DST=192.168.1.5 LEN=46 TOS=0x00 PREC=0x00 TTL=127 ID=60929\
 PROTO=ICMP TYPE=8 CODE=0 ID=T=eth0 SRC=192.168.2.99 DST=19\
2.168.1.5 LEN=46 TOS=0x00 PREC=0x00 TTL=127 ID=60929 PROTO=\
ICMP TYT=eth0 SRCT=eth0 SRC=192.168.2.99 DST=192.168.1.5 LE\
N=46 TOS=0x00 PREC=0x0T=eth0 SRC=192.168.2.99 DST=192.168.T\
=eth0 SRC=192.168.2.99 DST=192.168.1.5 LEN=46 TOS=0x00 PREC\
=0x00 TTLT=eth0 SRC=192.T=eth0 SRC=192.168.2.99 DST=192.168\
.1.5 LEN=46 TOST=eth0 SRC=192.168.2.99 DST=192.168.1.5 LEN=\
46 TOS=0x00 PREC=0x00 TTL=127 ID=6T=eth0 SRC=192.168.2.99 D\
STT=eth0 T=eth0 SRC=192.16T=eth0 SRC=192.168.2.T=eth0 SRC=1\
92.168.2.99 DST=192.168.1.5 LEN=46 TOS=0x00 PREC=0x00 TTL=1\
27T=eth0 SRC=192.168T=eth0 SRC=19T=eth0 SRC=192.168.T=eth0 \
SRC=192.168.2.99 DST=192.168.1.5 LEN=46 TOS=0x00 PREC=0x00 \
TTL=127 ID=60929 PROT=e
```

Gut zu wissen, dass Computer sich auch mal Fehler machen.

An diesen Stellen (so sieht fast das ganze Protokollfile aus) wurde immer nur eine Zeile gezählt.

4.5.2 Ping-of-Death Attack

Die Ping-of-Death-Attacke versucht durch ein übergroßes Paket einen Fehler im Netzwerkstack des Zielrechner auszunutzen und diesen so zum Absturz zu bringen.

Die Spuren dieser Angriffe waren schon schwieriger zu finden. Das öffnen neuer Verbindungen lief nebenbei ungestört weiter, genauso wie der Kommandointerpreter auf der Konsole. Es gab auch keine Aussetzer oder "Vermanschungen" im Logfile.

Die obere Zeile fand sich in ähnlicher Form nur vier und untere 14 mal im Log.

```
Jun 16 15:37:52 INRN05 kernel: NET: 24 messages suppressed.
Jun 16 15:37:52 INRN05 kernel: Oversized IP packet from 192\
.168.2.99.
```

4.5.3 Smurf Attack

Laut `http://www.computec.ch/software/denial_of_service/smurf/index-.html` sendet die Smurf-Attacke ein Ping mit gefälschte Absenderadresse (hier versehentlich die Falsche angegeben: 192.168.2.99) an eine Broadcastadresse (hier 192.168.2.255). Laut RFC haben alle Computer in diesem Netzwerk dieses Ping zu erwidern. Wird nun die Absenderadresse auf das Angriffsziel gesetzt, so erhält dieser Rechner <Anzahl der Attackten> mal <Anzahl der Rechner in diesem Netzwerk> viele Pings. In unserem Fall waren allerdings nur zwei (Mit Ping-Rechner drei) Rechner im Netzwerk, was die Attacke ins leere laufen lassen hat.

Zudem war die Source-Adresse mit 192.168.2.99 (192.168.2.100, die Firewall bzw. 192.168.1.4, der HTTP-Server wäre richtig gewesen) falsch angegeben. Mit dieser Einstellung griff sich der InterEmulator sozusagen selbst an.

Die Attacken beantwortet und so den eigentlichen Angriff hat in diesem Fall wohl der Ping-Rechner durchgeführt. Diesen hatten wir Zeitweise in das Netzwerk geschaltet um es mittels Pings den Fehler im Netzwerk zu suchen.

Die Firewall filterte alle Pings aus und konnte diese daher nicht mehr beantworten:

```
Jun 16 15:43:01 INRN05 kernel: DEFAULT DROP INPUT IN=eth1 O\
UT= MAC=ff:ff:ff:ff:ff:ff:00:04:55:04:40:00:08:00 SRC=192.1\
68.2.99 DST=192.168.2.255 LEN=46 TOS=0x00 PREC=0x00 TTL=128\
 ID=60929 PROTO=ICMP TYPE=8 CODE=0 ID=1 SEQ=1
Jun 16 15:43:01 INRN05 last message repeated 182 times
```

Die Attacke hatte also nur eine indirekte Wirkung auf den Transfer: Durch die Belastung des Netzwerkes durch den Angriff auf sich selbst ist der HTTP-Transfer etwas nach unten gegangen.

4.5.4 SYN Attack

Laut InterEmulator wurden keine Angriffe gesendet. Im Protokollfile waren ebenfalls keine Spuren eines Angriffs zu finden.

5 Anhang: Verwendete Geräte

5.1 Navtel InterEmulator

```
Navtel Comnunications Inc. IP InterEmulator Serial 000001394
```

Betriebsystem: Microsoft Windows NT

5.2 Firewall

- CPU Typ: AMD Athlon XP, 1544 MHz (5.75 x 269) 1800+
- Motherboard: Asus A7V266-EX (5 PCI, 1 AGP Pro, 1 ACR, 3 DIMM)
- Motherboard Chipsatz: VIA VT8366A Apollo KT266A
- Systembus: PCI-32 mit 33MHz
- Arbeitsspeicher: 128 MB (DDR SDRAM)
- BIOS Typ: Award Medallion (01/23/02)
- Grafikkarte: 3D Prophet 4000XT TV Out (32 MB)
- 3D-Beschleuniger: ST PowerVR Kyro
- Soundkarte: C-Media CMI8738 Audio Chip
- Diskettenlaufwerk

- Festplatte (40 GB, 7200 RPM, Ultra-ATA/100)
- ATAPI CD-R/RW 24X10 (24x/10x/40x CD-RW)
- PIONEER DVD-ROM DVD-116 (16x/40x DVD-ROM)

5.2.1 Netzwerkkarte (eth0) TP

- Hersteller: 3Com
- Typ: 3Com Etherlink XL 3C905C-TX-M
- Geschwindigkeit: 10/100 MBps
- Bus: PCI
- Funktionallität: IEEE 802.1p (Prioritätsvergabe für Datenübertragung)
- Sonstiges: Passive (d.h. kein Prozessor auf der Netzwekkarte)
- IP: 192.168.1.4
- MAC: 00:30:84:6C:3A:BE

5.2.2 Netzwerkkarte (eth1) LWL

- Hersteller: Allied Telesyn
- Typ: Allied Telesyn AT-2700FX-PCI-100MB-Ethernetadapter
- Geschwindigkeit: 10/100 MBps
- Bus: PCI
- Funktionallität: IEEE 802.1p (Prioritätsvergabe für Datenübertragung)
- Sonstiges: Passive (d.h. kein Prozessor auf der Netzwekkarte)
- IP: 192.168.2.100
- MAC: 00:04:76:DA:F6:BA

5.2.3 Betriebsystem

- Distribution: SuSE 9.0
- Kernel: `Linux 2.4.21-199-athlon #1 Fri Mar 12 08:24:04 UTC 2004 i686 athlon i386`
- Socks-Implementation: BSD Sockets