Grundpraktikum Netz- und Datensicherheit

Thema:
Konfiguration von Firewalls

Lehrstuhl für Netz- und Datensicherheit
Ruhr-Universität Bochum

Versuchdurchführung: Raum ID 2/168

Betreuung: Dominik Noß
Zusammengestellt von: Michael Psarros, Cornelia Menzel, Enrico Hartema, Florian Bache, Endres Puschner, Dominik Noß
Stand: 24. November 2021
Version: 4.1

1 Theorie von Firewallsystemen

1.0.1 Der Begriff Firewall

Das englische Wort „Firewall“ lässt sich mit „Brandschutzmauer“ übersetzen. Aufgabe einer solchen Brandschutzmauer ist es, das Feuer nicht hindurchzulassen, aber manchmal weist sie auch Löcher („Brandschutztüren“) auf. Diese erlauben ein Passieren bis ein Gefahrenfall („Brand“) eintritt.

Der 100%-ige Schutz gegen Gefahren aus dem Internet ist nur dann gegeben, wenn gar keine Verbindung besteht. Eine Firewall bietet an dieser Stelle einen Kompromiss. Sie möchten Daten über das Internet oder andere externe Netze mit anderen austauschen; gleichzeitig soll aber auch sichergestellt werden, dass nur die geforderten Anwendungen und der damit verbundene Datenaustausch realisiert werden kann. Andere Daten sollen keinesfalls in das eigene Netzwerk gelangen oder dieses verlassen.

In diesem Sinne ist eine Firewall eher vergleichbar mit einer Pforte an der Schnittstelle zwischen einem öffentlichen und einem privaten Netzwerk mit strenger Zugangskontrolle.

1.0.2 Was macht eine Firewall aus und was leistet sie?

Eine Firewall ist ein Ergebnis eines grundlegenden Sicherheitskonzeptes, indem wiederum Sicherheitsziele und Richtlinien festgelegt werden, um die sichere Verbindung mindestens zweier (Sub-)Netze zu gewährleisten.

Eine Firewall besteht in der Regel aus mehreren Komponenten. Dieser Versuch wird nur einige dieser Technologien abdecken, eine vollständige Implementierung einer Firewall zwischen großen Netzen sollte je nach Sicherheitskonzept eine Vielzahl dieser abdecken. In der folgenden Auflistung finden Sie einige dieser Technologien:

Paketfilterung Eine der Schlüsselfunktionen einer Firewall ist die Filterung von TCP/IP-Paketen. Hierbei werden eingehende Pakete überprüft und durch Abgleich mit einem Regelwerk durchgelassen (`accept`) oder verworfen (`drop`). Ebenfalls möglich ist das Verwerfen mit einer zusätzlichen Fehlerausgabe (`reject`). Beachten Sie, dass Beides auch eine Form der Information für einen Angreifer darstellt.

Network Adress Translation (NAT) Hierbei werden IP-Adressen durch andere ersetzt. Man unterscheidet dabei u.a. zwischen Source NAT (SNAT) und Destinaton NAT (DNAT). Bei Ersterem werden die Quelladressen durch nicht benutzte öffentliche IP-Adressen ersetzt, beim Letzteren die Zieladresse. Da man oftmals durch NAT private Adressen verstecken möchte, spricht man auch von IP-Masquerading.

Proxy Um auf der Anwendungsschicht des TCP/IP-Referenzmodelles ebenfalls Sicherheit zu gewährleisten, können Proxies (Englisch für Stellvertreter) genutzt werden. Diese agieren stellvertretend für die Clients in einem Netzwerk, indem sie Anfragen weiterleiten, können aber auch stellvertretend für interne Server agieren, indem sie zunächst besagte Anfragen bearbeiten und inspizieren.

Content-Filtering Um den Inhalt von Paketen im Rahmen der Anwendungsschicht zu analysieren und ggf. zu verwerfen, werden Content-Filter benutzt. Anfragen, die beispielsweise pornographische Inhalte und/oder Signaturen von Viren enthalten, können so effektiv geblockt werden. Nutzt eine Firewall Content-Filtering und/oder Proxies, spricht man auch von einer *Application Layer Firewall* (ALF).

Authentifizierung Durch das Nutzen von starker Authentifizierung können Firewalls die Möglichkeit bieten, bestimmte Dienste erst nach erfolgreicher Authentifikation zu offerieren.

Intrusion Detection/Prevention Systeme (IDS/IPS) Auf einer Firewall können IDS bzw. IPS aufsetzen. Ein IDS sucht den Datenverkehr nach Angriffen ab und protokolliert diese, ein IPS führt nach Erkennung eines Angriffes entsprechende Gegenmaßnahmen aus.

1.0.3 Was leistet eine Firewall nicht?

Firewalls schützen nur vor Verbindungen, die über sie laufen Die Zugangskontrolle ist nur so stark, wie ihr schwächstes Glied. Möchte man den Datenverkehr aus einem Unternehmen heraus unterbinden, so nützt eine Firewall wenig, wenn die Mitarbeiter mit USB-Sticks oder mobilem Internet (UMTS) am Arbeitsplatz hantieren können. Ein Administrator sollte diese Möglichkeit so gut wie möglich unterbinden (bspw. Deaktivieren von nicht genutzten USB-Anschlüssen bzw. COM-Ports und Setzen von starken BIOS-Passwörtern. Aber auch das hilft nicht gegen ein Reset des BIOS)

Angriff von innen Wenn sich jemand erlaubte Kanäle zu nutze macht, kann die Firewall umgangen werden. Dies führt zu neuen Angriffsszenarien: trojanische Pferde setzen sich im internen Netzwerk fest und nutzen die „Restdurchlässigkeit“ der Firewall, um Daten nach außen zu transferieren.

Erlangt ein Angreifer physikalischen Zugang zu dem internen Netzwerk, so kann die Firewall nichts gegen einen solchen Angriff ausrichten; sie kontrolliert nur die Schnittstelle zwischen internem und externem Netzwerk.

Tunneln Mit Hilfe von Tunneln gebraucht man Protokolle für artfremde Dateien/andersartige Protokolle. Je nach Firewallspezifikation können diese nicht erkannt werden, da sie für das Regelwerk als valide erachtet werden. Beliebt zum Umgehen einer Firewall ist beispielsweise das Tunneln über DNS.

1.0.4 Typischer Firewall-Aufbau

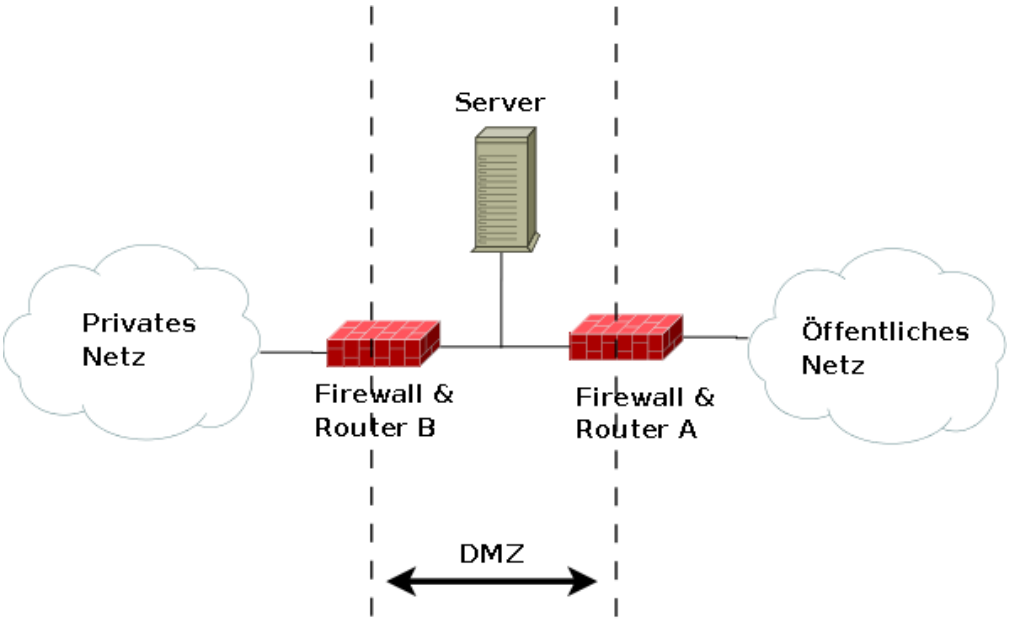

Abbildung 1.1: Typischer Firewall-Aufbau mit DMZ

Zu erkennen ist der typische Aufbau einer zweistufigen Firewall. Das öffentliche Netz wird dabei mit Hilfe der Firewall A von der DMZ (Demilitarisierte Zone) abgeschirmt. In der DMZ befinden sich Server, welche aufgrund ihrer Aufgaben dem öffentlichen Netz Dienste zur Verfügung stellen müssen, beispielsweise ein Webserver. Da dieser Server besondere Eigenschaften aufweist und besonders geschützt werden sollte, wird dieser auch Bastion-Host genannt. Um das private Netz durch diesen Zugang nicht zu gefährden, integriert man eine weitere Firewall, welche das private Netz wiederum von der DMZ abschirmt.

Dieser Aufbau ist nur ein Beispiel, es gibt zahlreiche verschiedene Aufbaukonzepte für Firewalls. Das Konzept einer zweistufigen Firewallkonstruktion mit einer DMZ ist jedoch typisch, so wird dieses auch vom BSI in seinen IT-Grundschutzkatalogen [1] empfohlen.

Der Übersichtlichkeit halber sind Router und Firewall gemeinsam im Bild integriert, grundsätzlich sollte man jedoch darauf achten, die Systeme so autonom wie möglich zu halten. Jedes System sollte nur die eine Aufgabe erfüllen, für die es auch gedacht ist, weitere zusätzliche und unnötige Funktionen können die Sicherheit beeinträchtigen. Das gilt für Router, Firewalls (explizit Paketfilter) und auch Server.

1.0.5 Sicherheitsziele in einem Firewallkonzept

- negative default Policy: „Alles, was nicht ausdrücklich erlaubt ist, ist verboten“
  Sogenannte „default Policies“ sind Regeln, die sich auf alles beziehen, was nicht ausdrücklich spezifiziert ist. Eine negative default Policy bietet deutlich mehr Sicherheit als eine positive und sollte deswegen vorgezogen werden.
- Zugriff auf interne Anwendungen durch autorisierte Dritte:
  Möchte man zum Beispiel einen Heimarbeitsplatz einrichten, so muss der Angestellte z.B. über das Internet auf die internen Server zugreifen können. Ähnliches ist denkbar, wenn im Firmennetz z.B. ein Online-Shop betrieben wird, der für alle erreichbar sein soll. Ein entsprechender Server sollte in die DMZ eines Netzwerkes integriert werden.
- Einschränkung der Internetbenutzung der Angestellten:
  Benötigen einige Anwender an ihrem Arbeitsplatz Zugriff auf das Internet, so ist es ratsam dafür zu sorgen, dass nur benötigte Dienste genutzt werden können und andere (z.B. Filesharing) nicht. Es sollte hier abermals betont werden, dass eine bloße Diensteinschränkung mit Hilfe eines Paketfilters nicht ausreicht. So laufen mittlerweile viele Filesharing-Programme auf dem HTTP-Port (80), hier sollte man durch Content-Filtering es dem Mitarbeiter zumindest erschweren, diese und ähnliche Dienste ausführen zu können.

1.0.6 Netzwerkprotokolle und Grundlagen TCP/IP

Firewalls sind natürlich abhängig von dem zugrundeliegenden Protokoll. Wir beschränken uns hier auf IPv4, welches immer noch eine rege Verbreitung besitzt. Die in diesem Versuch erlernten Methoden zur Konfiguration von Firewalls gelten allerdings prinzipiell auch für IPv6.

Um eine Firewall erfolgreich konfigurieren zu können, sind gewisse Grundkenntnisse bezüglich dieser Protokolle notwendig. Für weitere Informationen wird die Vorlesung „Computernetze“[6] empfohlen.

Die ISO (International Standards Organization) hat zum Vergleich von Netzwerkprotokollen und zur Umsetzung eines modularen Konzeptes das OSI-Modell (Open Systems Interconnect) entwickelt, welches in Abbildung 1.2 links zu sehen ist. Dieses besteht aus sieben Schichten („Layer“), die alle aufeinander aufsetzen. In der Skizze sind sowohl die deutschen als auch die englischen Begriffe abgebildet. Die unterste Schicht kümmert sich dabei um die physikalische Übertragung, nach oben hin wird das Modell

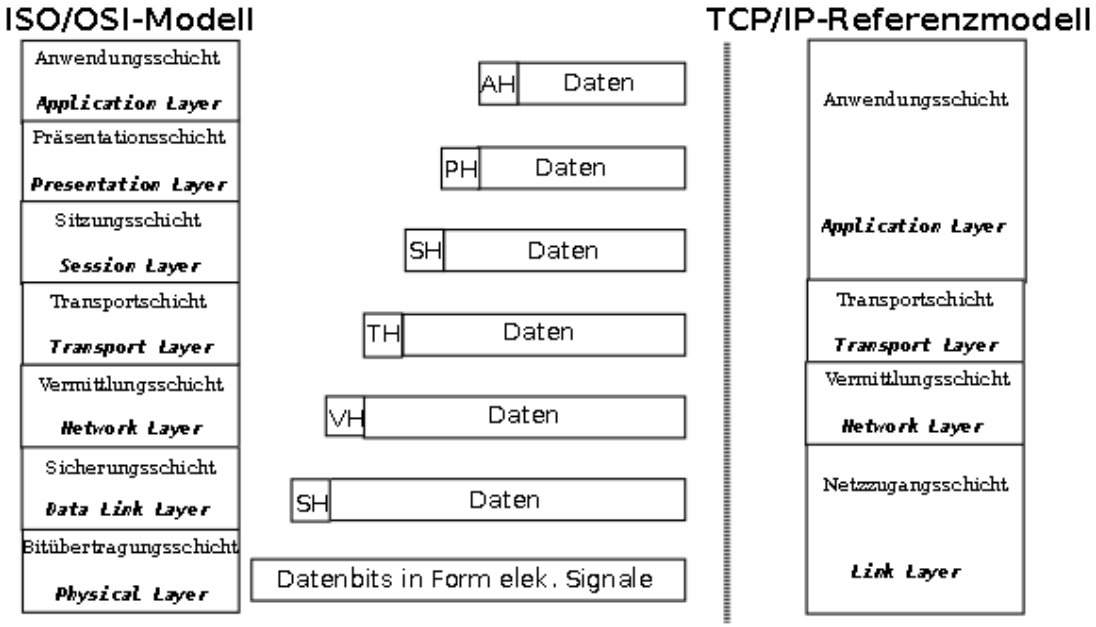

Abbildung 1.2: Darstellung des ISO/OSI- und TCP/IP-Referenzmodells

allgemeiner bis hin zur Anwendungssschicht (nicht zu verwechseln mit dem eigentlichen Anwendungsprozess), die es einem Benutzer/Programm erst ermöglicht, eine Netzwerkübertragung einzuleiten. Da dieses Modell für das eigentliche Netzwerkprotokolldesign gedacht ist, existiert für die Internetprotokollfamilie ein etwas weniger detailliertes Modell, das sogenannte TCP/IP-Referenzmodell, welches Schichten des ISO/OSI-Modelles zusammenfasst. Wenn eine Nachricht im Rahmen einer Netzwerkumgebung von einem Rechner zu einem anderen geschickt wird, durchläuft es mehrere (nicht unbedingt alle) Schichten der Modelle. Ein Programm oder eine Netzwerkkomponente kann dabei die Nachricht auswerten, benötigte Informationen stehen dabei im Header (Kopf) der Nachricht. Da jede Schicht durch individuelle Informationen geprägt wird, existieren mehrere Header. So wertet beispielsweise ein Switch, der im Allgemeinen auf der Sicherungsschicht (Data Link Layer) operiert, nur die Headerinformationen auf dieser Schicht aus (bspw. MAC-Adresse), nachfolgende Header werden als Daten aufgefasst. Dieser Verlauf von Daten ist in der Mitte der Abbildung zu sehen. Wie man erkennen kann, wird die Nachricht durch die zusätzlichen Header in Richtung der unteren Schichten größer. Zu beachten ist jedoch, dass die abgebildeten Verhältnisse dabei keinesfalls praxisnahe Größen widerspiegeln. Vorherige Header wurden bereits vorher abgearbeitet und der Nachricht entnommen. Wir wollen nun im Folgenden die einzelnen Schichten des Referenzmodells im Detail betrachten.

Netzwerkschicht Die Netzwerkschicht bezieht sich auf die physikalische Verkabelung (im Netzlabor: Ethernet 100Base-T) und ist für den Versuch nicht relevant. Es existieren jedoch auch Paketfilter, mit denen man auf der zweiten Schicht des ISO/OSI-Modells (also dem Data Link-Layer) filtern kann. So kann man den Zugriff von bestimmten MAC-Adressen kontrollieren, das Fälschen von diesen Adressen ist jedoch kein Problem.

Internetschicht Nun betrachten wir einen Teil der Internetschicht, IP. Wenn ein IP-Datagramm einen Router passiert, bleibt der Header weitestgehend erhalten. Die TTL wird jedoch beim Passieren eines Routers dekrementiert. Dadurch muss die Prüfsumme ebenfalls neu berechnet werden. Auch Optionen und QoS-Elemente können gegebenenfalls neu gesetzt werden.

Wichtige Elemente des IP-Headers sollen nun erläutert werden. Im Feld „protocol“ wird angegeben, welches Protokoll der nächsthöheren Schicht in dieses IP-Paket eingepackt wurde. In den Feldern „source ip address“ und „destination ip address“ steht die Adresse des Senders und Empfängers. Die IP-Adresse setzt sich aus der Netzwerkadresse und der Hostadresse zusammen.

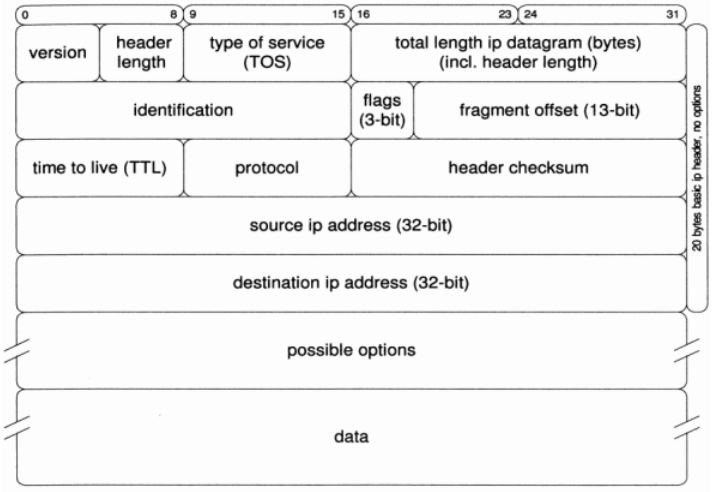

Abbildung 1.3: Header eines Pakets im Internet Protocol (IP)

Mit Hilfe der Subnetzmaske[9] definiert man den Anteil des Netzpräfixes, das heißt, wie groß das Netz ist. Möchte man Verbindungen zwischen zwei Netzwerken herstellen, benötigt man einen Router. Beispiele zu Netzadressen sind in Tabelle 1.1 zu finden.

Netzadresse	**Subnetzmaske**	**Erster Host**	**Letzter Host**	**Broadcast**
192.168.0.0	255.255.255.0	192.168.0.1	192.168.0.254	192.168.0.255
172.16.0.0	255.255.0.0	172.16.0.1	172.16.255.254	172.16.255.255
134.147.20.0	255.255.255.128	134.147.20.1	134.147.20.126	134.147.20.127

Tabelle 1.1: Beispiele zu Netzadressen

Die Subnetzmasken werden gemäß CIDR[2] betrachtet, eine alternative und zeitsparende Schreibweise existiert ebenfalls. So lässt sich 255.255.255.0 auch als /24 beschreiben. Dabei betrachtet man 255 als Binärzahl (11111111) und summiert die Anzahl der Einsen auf. Da 255 drei Mal vorhanden ist, ergibt sich ein Wert von $3 \cdot 8 = 24$.

Transportschicht Auf der Internetschicht aufsetzend sorgt die Transportschicht für die Segmentierung von Nachrichten. Applikationen bekommen durch sie eine logische Ende-zu-Ende-Übertragung bereitgestellt. Applikationen können durch Port-Nummern identifiziert werden. Diese Nummern befinden sich ebenfalls in den Headern von Protokollen der Transportschicht. Es sollen nun drei Protokolle dieser Schicht vorgestellt werden:

- TCP (Transmission Control Protocol)
  TCP ist ein verbindungsorientiertes Protokoll und wird in der Transportschicht implementiert. Es transportiert keine Datenblöcke, sondern Datenströme; man spricht also von einem TCP-Stream. TCP besitzt eine weitere Adressierung, die so genannte Port-Nummer. Während die IP-Adressierung für die Zustellung zum richtigen Host sorgt, dient die Portadressierung der Auswahl des richtigen Dienstes. Zunächst sendet der Client von Port 1025 aus ein SYN-Paket (synchronisation) an den Port 80 (HTTP) des Servers. Dieser bestätigt den korrekten Erhalt mit einem ACK (acknowledgement) und initiiert seinerseits die Verbindung mit einem SYN. Diesen Verbindungsaufbau nennt man auch TCP-Handshake. Danach können die beiden Rechner Daten austauschen. Die Pakete verfügen dann über eine eindeutige Sequenznummerierung, die die Vollständigkeit der übertragenen Daten sicherstellt. Der Zustand nach dem Aufbau einer TCP-Verbindung lässt sich auch als zustandsbehaftet beschreiben.
- UDP (User Datagramm Protocol)

UDP funktioniert im Prinzip wie TCP, ist jedoch zustandslos und verzichtet damit auf das Handshake. Aufgrund des reduzierten Overheads ist UDP ein beliebtes Protokoll bei Anwendungen, die eine reduzierte Verzögerungszeit benötigen, beispielsweise bei der Internettelefonie.

- ICMP (Internet Control Message Protocol)
  ICMP wird oft als Bestandteil der Netzwerkschicht angesehen, wird aber wie TCP und UDP über IP-Pakete versendet. In jedem Netzwerk können Fehler oder andere unvorhergesehene Zustände auftreten. Solche Zustände werden mit ICMP-Nachrichten gemeldet. Anstelle von Portnummern besitzt ICMP Nachrichtentypen, die den Inhalt der Nachricht beschreiben. U.a. die Programme `ping` und `traceroute`[1] nutzen ICMP.

1.0.7 Zustandslose und zustandsbehaftete Paketfilterung

Verbindungsorientierte Protokolle wie TCP haben nach dem Aufbau einen Zustand, den man als „established“ bezeichnet. Mit diesem Wissen können wir Paketfilter differenzieren und zwischen zustandslosen und zustandsbehafteten Paketfiltern unterscheiden. Erstere, im Englischen auch als „stateless“ bezeichnet, arbeiten unabhängig von bereits eingegangen Paketen, was im Widerspruch zur Netzwerkschicht steht (siehe TCP). Zustandsbehaftete Paketfilter, auch als „stateful“ betrachtet, können sich hingehend dem Netzwerkverkehr anpassen. Pakete, die zu genau einer bestehenden Verbindung gehören, können so eindeutig zugeordnet werden.

Im Deutschen sind auch in der allgemeinen Literatur die Begriffe statisch und dynamisch üblich, jedoch versteht man beispielsweise unter dynamischen Paketfiltern eher jene, die durch Interagieren mit einem Intrusion Detection System während des Betriebes neue Regeln generieren. Diese Regeln entstammen der Analyse des Netzwerkverkehres durch das IDS.

1.1 Praxis von Firewallsystemen

Unter Linux bietet `netfilter/iptables` [3] ein umfangreiches Paketfiltersystem, das Sie in diesem Praktikumsversuch kennenlernen sollen.

Der Kernel arbeitet in der Paketfiltertabelle mit Listen („Ketten“) von Regeln. Diese Listen bezeichnet man als „Firewall Chain“. Die einzelnen Filterregeln können jederzeit in die Regelkette eingefügt oder entfernt werden. Mit dem Tool iptables können auch zur leichteren Verwaltung und besseren Lesbarkeit eigene Chains definiert werden.

Den Ketten übergeordnet sind die Tabellen, „Tables“ genannt. Jede Funktionsart der Firewall (Paketfilterung: filter, Network Address Translation: nat) wird in einer Tabelle zusammengefasst. Zunächst werden die Funktionen der Tabelle „filter“ betrachtet.

1.1.1 Ablauf des Filtervorganges

Wenn ein Paket über ein beliebiges Interface in das System hereinkommt, wird ein vorgezeichneter Weg durchlaufen. In der „routing decision“ wird überprüft, ob das Paket für den PC selbst oder zur Weiterleitung in ein anderes Netzwerk beim Routerbetrieb bestimmt ist.

In jeder Regelkette werden die einzelnen Regeln der Reihenfolge nach abgearbeitet und geprüft, ob eine Regel auf das aktuelle Paket zutrifft oder nicht. Trifft eine Regel zu, wird – vereinfacht gesagt – das Paket akzeptiert oder abgelehnt. An dieser Stelle wird dann die jeweilige Kette verlassen, es sei

[1] Programm, welches die Router, über die eine Nachricht gesendet wird, ermitteln kann

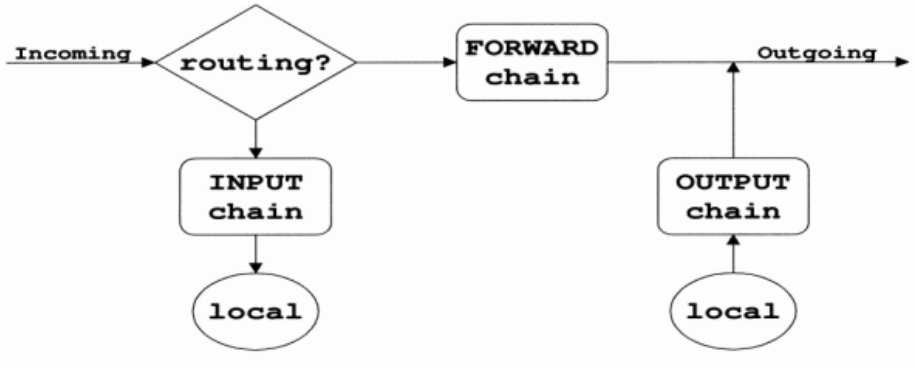

Abbildung 1.4: Ablauf des Filtervorganges in Netfilter

denn, die Regel dient ausschließlich der Protokollierung. Deshalb ist die Reihenfolge der Filterregeln bei der Implementierung sehr wichtig. Sind alle Regeln einer Kette durchlaufen, ohne dass das Paket zugeordnet werden konnte, greift am Ende der Verarbeitung die Default Policy.

1.1.2 Filterregeln

Filterregeln lassen sich grob in drei Bestandteile zerlegen:

- die Grundoperationen, wie Einfügen einer Regel, Regelketten anlegen und löschen,
- „Matching"-Optionen entscheiden, ob die Regel überhaupt auf ein Paket zutrifft oder nicht und
- „Targets", die beschreiben was mit einem Paket geschehen soll, wenn die Regel zutrifft.

Somit ergibt sich die Syntax des Aufrufs:

- `iptables [-t table] -[I,A,D] [chain] rule-specification [options]`
  fügt eine Filterregel am Beginn einer Regelkette (insert) oder am Ende (add) oder löscht die spezifizierte Regel (delete)
- `iptables [-t table] -L [chain] [options]`
  zeigt alle oder die angegebenen Ketten der Tabelle an
- `iptables [-t table] -P [chain] target`
  legt die „default policy" der ausgewählten Kette fest
- `iptables [-t table] -F`
  löscht alle Regeln der ausgewählten Kette oder aller Ketten der angegebenen Tabelle

Standardmäßig wird die Tabelle filter verwendet, es sei denn, man gibt explizit eine andere Tabelle an. Mögliche Werte für *`table`* sind `filter` und `nat`; bei *`chain`* können `INPUT`, `OUTPUT`, `POSTROUTING` usw. angegeben werden (letztere nur in Verbindung mit nat). In dem Versuch sollten Sie stets die Option `-n` als Parameter im Bereich *`options`* angeben, um eine numerische Ausgabe der IP-Adressen zu erzwingen.

1.1.3 Konstruktion der Regeln

Die *rule-specification* setzt sich aus den Matching Optionen und einem Target zusammen:

- `rule-specification = matching1 [matching2 [...]] -j target`

Durch das Matching wird ausgewählt, auf welche Pakete die Regel überhaupt zutreffen soll. Es ist eine Hintereinanderreihung der im Folgenden besprochenen Parameter (zu trennen mit einem Leerzeichen). *`target`* taucht auch bei der Syntax der Default Policy auf und legt fest, was mit den Paketen geschehen

soll.

1.1.4 Matching Optionen

- `-p` *protocol*
  Mögliche Angaben für protocol sind TCP, UDP und ICMP.
- `-s IP`
  Quelladresse; es können auch Adressbereiche angeben werden in der Schreibweise „Netzadresse/Subnetzmaske“.
- `-d IP`
  Zieladresse
- `-i interface` und `-o interface`
  Alle Interfaces können hier als Ein- bzw. Ausgangsinterface angegeben werden. Die im System verfügbaren Interfaces werden mit `ip l` angezeigt.
- Negation: `!`
  Alle Matches können durch ein Ausrufezeichen vor dem Argument negiert werden.

1.1.5 Protokollspezifische Erweiterungen der Matching Optionen

Mit dem Argument `-p tcp` werden weitere Funktionen nachgeladen, die eine feinere Konfiguration in Bezug auf TCP ermöglichen.

- `--source-port Portnummer` und `--destination-port Portnummer`
  Es können einzelne Ports angegeben werden. Mit dem Modul multiport sind auch ganze Portbereiche („6000:6015“) möglich.
- `-m state --state [!] Zustand`
  Ermöglicht den Zustand von Verbindungen zu kontrollieren. Dabei werden die Daten ausgewertet, die das Connection-Tracking-Modul sammelt. Zulässige Werte für Zustand sind dabei `NEW` (das Paket initiiert eine Verbindung), `ESTABLISHED` (das Paket gehört zu einer bestehenden Verbindung), `RELATED` (Pakete, die in Beziehung zu einer offenen Verbindung stehen, aber nicht direkt zu dieser Verbindung gehören) und `INVALID` (Pakete, die nicht genau identifiziert werden können).

Vergleichbare Funktionen mit Ausnahme der Prüfung auf Verbindungszustand sind auch bei UDP möglich. Geben Sie hierzu `-p udp` an.

1.1.6 Targets

Trifft eine Regel auf ein Paket zu, so wird das Paket dem Target zugeleitet. Es existieren unter Anderem zwei triviale Targets:

- `ACCEPT`
  Das Paket wird durchgelassen
- `DROP`
  Das Paket wird verworfen

Außerdem ist es möglich, eigene Chains zu entwerfen, an die das Paket dann weitergeleitet wird.

Beispiel:

Ein PC2 (IP 192.168.1.2) soll auf Port 8080 den PC3 (IP 192.168.1.3) ansprechen können. Hierbei

sollen Pakete, die zur bestehenden Verbindung gehören, akzeptiert werden. Auf einem Router, der das Koppelelement zwischen den beiden PCs ist und alle Pakete standardmäßig verwirft, müssen daher folgende Regeln implementiert werden:

```
iptables -t filter -A INPUT -s 192.168.1.2 -d 192.168.1.3 -p tcp --dport 8080 -j ACCEPT
iptables -t filter -I OUTPUT -m state --state ESTABLISHED -j ACCEPT
```

1.1.7 NAT

Wie schon erwähnt, handelt es sich bei NAT um eine Adressumsetzung, die mit Hilfe von iptables an drei verschiedenen Stellen geschehen kann, siehe Abbildung 1.5.

NAT kennt drei verschiedene Regelketten: `PREROUTING`, `POSTROUTING` und `OUTPUT`. Letztere ist für den Versuch nicht erforderlich. Zwar sind Paketfilterung und NAT unabhängig voneinander, aber die Reihenfolge, in der ein Paket die einzelnen Regelwerke durchläuft, spielt bei der Konstruktion der Filterregeln eine wichtige Rolle.

Die Regelkette `PREROUTING` kommt bei DNAT (für „destination“) zum Zuge und wird durchlaufen, sobald das Paket eingetroffen ist. Die Routing Decision erfolgt erst danach.

`POSTROUTING` schließlich ist die Regelkette für alle SNAT (für „source“) Anwendungen und wird erst durchlaufen, wenn das Paket ins Netzwerk gesendet wird. In diesem Versuch beschränken wir uns auf einen Spezialfall des NAT, wo ein internes Netzwerk hinter einer IP eines Weitverkehrsnetzes versteckt wird. Hierzu verwendet man einen Eintrag in der POSTROUTING-Tabelle mit dem Target `MASQUERADE`. Bei der Konstruktion des entsprechenden Eintrages muss das abgehende Interface (also das Interface, hinter dessen IP das zu schützende Netzwerk gegenüber dem großen Netz versteckt wird) mit angegeben werden. Natürlich erfolgt die Zusammenstellung der *rule-specification* genau wie bei der Tabelle filter.

Eine gute Einführung in die Funktionsweise dieses Firewallsystems bietet [5] im Kapitel 3, 4 und 6 und die offizielle Dokumentation [4].

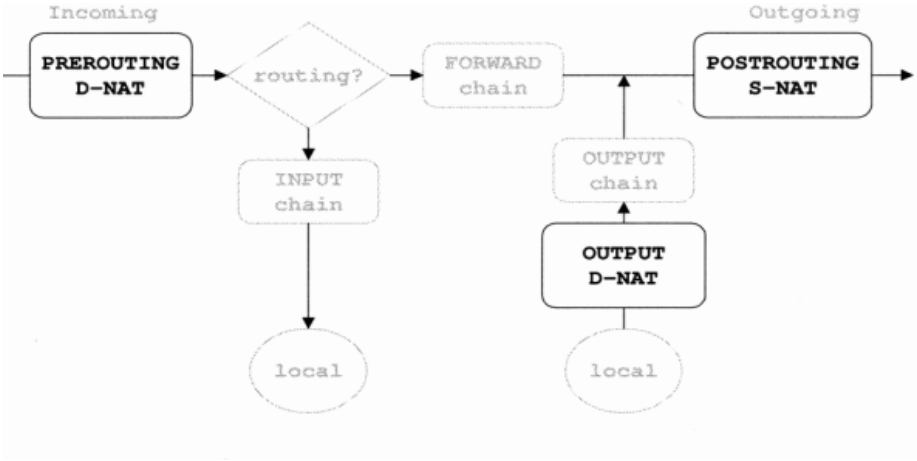

Abbildung 1.5: Ablauf des NAT-Routings von netfilter

1.1.8 String Matching

Seit dem Linux-Kernel 2.6.18 ist es möglich, mithilfe von iptables zumindest in kleinem Maße Content-Filtering zu betreiben. So ist es möglich, innerhalb von IP-Datagrammen nach Strings zu suchen. Dies erlaubt das Erstellen von Regelwerken, die bestimmten Inhalt (beispielsweise URLs oder Dateinamen

wie Trojaner.exe) filtern. Die Regeln fallen unter protokollspezifische Erweiterungen der Matching-Optionen, sollen hier aber nochmals gesondert aufgeführt werden:

- `-m string --string "<zu filternder string>" --algo bm|kmp`

Der Parameter „algo“ muss spezifiziert werden und gibt vor, welcher String-Matching-Algorithmus verwendet wird. Zur Auswahl stehen Boyer-Moore (bm) und der Knuth-Morris-Pratt-Algorithmus (kmp). Dabei kann der Erstere durch eine unter Umständen sublineare Laufzeit überzeugen, während Letzterer eine lineare Laufzeit garantiert. Für den Versuch ist der Boyer-Moore-Algorithmus, also der Parameter bm, vollkommen ausreichend, bei Interesse sei jedoch auf [10] verwiesen.

Es existieren weitere Parameter, mit denen man Offsets angeben kann, in denen gesucht werden soll, weiterhin ist die Suche nach Hexstrings ebenfalls möglich.

Beispiel:

Ein Webserver Ihrer Abteilung ist einem DDoS-Angriff [7] ausgesetzt. Nachdem Sie die Logs des Webservers untersucht haben, fällt Ihnen auf, dass nur eine bestimmte Domain angegriffen wird. Durch das Filtern dieser Domain ist der Zugriff auf den Webserver weiterhin möglich, die entsprechende Domain jedoch wird nicht mehr erreichbar sein. Dieses erreichen Sie durch den Befehl:

- `iptables -I INPUT -p tcp --dport 80 -m string --string "angegriffene-domain.de" --algo bm -j DROP`

Beachten Sie, dass hier ganz bewusst das bestehende Regelwerk durch den Parameter `-I` ergänzt wird. Diese Regel soll nämlich vor der eigentlichen akzeptierenden Regel liegen, die den Zugriff auf den Webserver gestattet.

2 Hinweise

2.1 Voraussetzungen für die Teilnahme

- Grundkenntnisse zum Arbeiten mit Linux und seiner Kommandozeile
- Die Versuchsanleitung muss gelesen und verstanden werden. Ferner müssen Sie die Kontrollfragen beantworten können.

2.2 Schriftliche Versuchsauswertung

Jedes Team fertigt eine schriftliche Auswertung an. In dieser müssen alle Kommandos und deren Ausgaben dokumentiert werden. Fertigen Sie ggf. Screenshots an. Außerdem sollten Sie Ihre einzelnen Schritte begründen und die Firewallregeln erklären.

Bitte geben Sie auch Feedback, ob Sie den Praktikumsversuch als interessant empfunden haben und ob dieses Dokument für Sie bei der Versuchsdurchführung hilfreich war. Verbesserungsvorschläge sind willkommen!

Die Versuchsauswertung ist in moodle vor dem nächsten Versuch abzugeben!

2.3 Versuchsbeschreibung

Wichtig: Um unnötige Fehler zu vermeiden, lesen Sie sich bitte alle Aufgaben und Erläuterungen langsam und gründlich, möglichst schon vor dem Versuch, durch!

Die neu gegründete Firma „Narf-Fnord“ hat aus logistischen Gründen seine ersten drei Sitze über Deutschland – genauer Berlin, Hamburg und München – verteilt. Zusätzlich wurde ein IT-Spezialist (Sie) als Administrator des Firmennetzes eingestellt, der von Bochum aus die Firma an das Internet anbindet und für die Netzwerkkonfiguration zuständig ist.

In der folgenden Abbildung lässt sich der bisherige Aufbau des Netzwerkes erkennen:

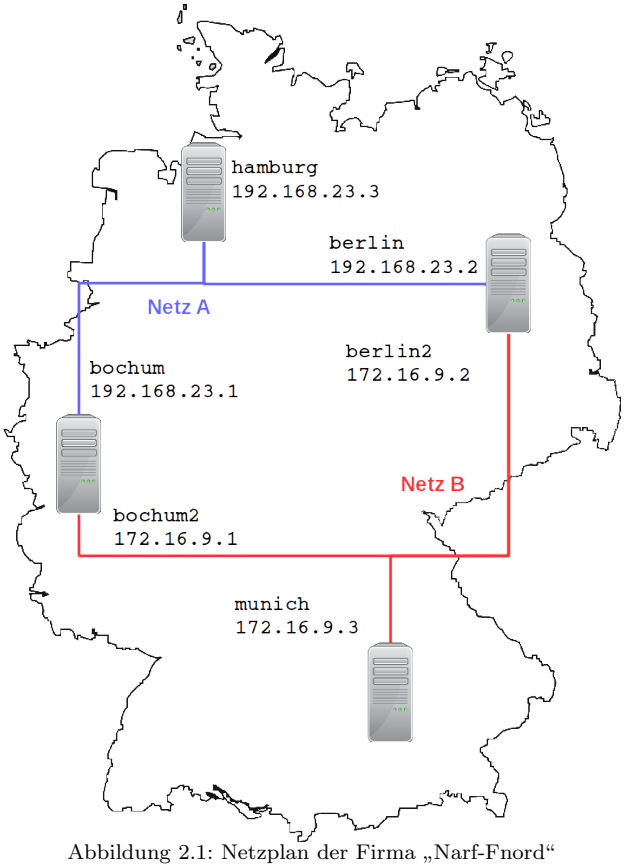

Abbildung 2.1: Netzplan der Firma „Narf-Fnord“

Das Netzwerk weist also folgende Eigenschaften auf: Bochum, München und Berlin bilden das Subnetz 172.16.9.0/24. Weiterhin bilden Bochum, Hamburg und Berlin das Subnetz 192.168.23.0/24. Beide Netze besitzen also die Subnetzmaske /24 gemäß CIDR.

Beachten Sie, dass Berlin und Bochum an beide Netzwerke angebunden ist, also über zwei Netzwerkadapter verfügt.

2.4 Wichtige Hinweise

Zugangsdaten:

- Ihr Arbeitsplatzrechner soll den host Bochum darstellen.
  Zugangsdaten: **student** / **12345**
- Berlin, Hamburg, Munich sind virtuelle Maschinen.
  Zugangsdaten: **student** / **12345** sowie **root** / **12345**

Host names:
Innerhalb der VMs ist es möglich, die Netzwerkadapter über ihre hostnames anzusprechen. So ist es z.B. möglich, ”**ping bochum**” durchzuführen. Dies ist äquivalent zu ”**ping 192.168.23.1**”.
So ist ”**ping bochum2**” äquivalent zu ”**ping 172.16.9.1**”. Alle hostnames sind kleingeschrieben.

Sie müssen nichts weiter konfigurieren. Wählen Sie aus der Menüleiste links einfach die zu startende virtuelle Maschine aus und klicken Sie in der oben Menüleiste auf „Starten“ (symbolisiert durch einen grünen Pfeil).

Wichtig ist die Tastenkombination **Strg Rechts+F**, mit der Sie den Vollbildmodus der virtuellen

Maschine beenden können. Bitte merken Sie sich diese unbedingt.

Als Basissystem dient innerhalb der virtuellen Maschinen die Linux-Distribution Debian.

Sie sollten in diesem Versuch bereits mit den Grundlagen beim Arbeiten in der Kommandozeile vertraut sein. Des Weiteren werden Sie mit iptables arbeiten, welches Ihnen in der obigen Einführung vorgestellt wurde.

Die zur Verfügung stehende VM enthält darüber hinaus eine Auswahl von Werkzeugen und Programmen (u.a. den Browser links, traceroute, host, sowie ping) zur Überprüfung der Konnektivität. Neben dem Webserver Apache2 setzen Sie in diesem Versuch auch OpenSSH als SSH-Server ein. Die Versionsnummern sind den einzelnen Programmen zu entnehmen.

3 Aufgabenteil

In allen Teilaufgaben wird von Ihnen erwartet, dass Sie die notwendigen Schritte dokumentieren und in ausreichender Genauigkeit in Ihrer Auswertung angeben. Nennen Sie bei redundanten Schritten Abweichungen oder Besonderheiten. Lesen Sie die Aufgabenstellung aufmerksam und bearbeiten Sie die Aufaben nacheinander.

3.0.1 Start der VMs

Sie sind auf Ihrem Arbeitsplatzrechner mit dem Benutzernamen „student“ angemeldet. Starten Sie die Virtualisierungssoftware VirtualBox über das Start-Menu oder mittels Konsolenbefehl "**virtualbox**". Starten Sie die drei VMs Berlin, Hamburg und Munich.

3.0.2 Start des Webservers

Prüfen Sie zunächst, ob der Webserver auf Berlin gestartet ist. Dies erreichen Sie durch `/etc/init.d/apache2 status`. Falls nötig, starten Sie den Dienst. Sollte der Dienst bereits laufen, erklären Sie, wie Sie ihn starten könnten.

3.0.3 IP-Forwarding einschalten

Nutzen sie eine Suchmaschine um den Befehl zum aktivieren von `ip forwarding` unter Linux herauszufinden. Es gibt mindestens zwei Varianten: eine mit `sysctl` und eine andere, bei der eine `1` in eine Datei geschrieben wird. Notieren Sie den Befehl in Ihrem Versuchsprotokoll.

3.1 Aufgabe 1: Erstes Einstellen der Firewall

Als erstes werden Sie mit der filter-Tabelle, speziell mit den Ketten INPUT und OUTPUT vom Standort Berlin, arbeiten.

Sie sollen in dieser Aufgabe insgesamt drei Rechner verwenden: Berlin, Hamburg und Ihren Arbeitsplatzrechner. Berlin dient als Server; Hamburg und Ihr Arbeitsplatzrechner als Client.

Bei der Einrichtung von Firewalls muss vor der Implementierung ein Regelwerk geplant werden. Sie werden in den folgenden Aufgaben ein solches Regelwerk planen.

Geben Sie die Regeln erst in iptables ein, wenn Sie dazu aufgefordert werden.

3.1.1 Aufgabe 1.1

Geben Sie die derzeitige Firewall-Konfiguration vom Standort Berlin für die Ketten INPUT und OUTPUT aus.

3.1.2 Aufgabe 1.2

Wählen Sie nun die „default policy“ für INPUT und OUTPUT begründen Sie ihre Wahl. Es soll *maximale Sicherheit* im Bezug auf ein- und ausgehende Verbindungen garantiert werden.

3.1.3 Aufgabe 1.3

Lesen Sie sich nun folgende Anforderungen *genau* durch:

Die Managementabteilung betrachtet die Firewalleinstellungen im Standort Berlin als zu sicher und möchte ein System haben, mit dem die Firma produktiv arbeiten kann. Es soll möglich sein, dass jeder die Dienste des Webservers nutzen kann; das heißt, die Webseite der Firma soll von jedem erreichbar sein, um über die wichtigsten Neuigkeiten informiert zu werden. Außerdem soll es möglich sein vom Standort Hamburg eine sichere Kommunikation nach Berlin aufzubauen, um sehr wichtige, firmeninterne Daten ebenfalls der Hauptzentrale mitteilen zu können. Hierbei soll ssh verwendet werden.

Füllen Sie zunächst die Tabelle aus Sicht von Berlin vollständig aus. Betrachten Sie dabei jeweils die ein- und ausgehende Richtung der beiden Verbindungsarten.

Nr.	Source	Destination	Protocol	sport	dport	Chain	Action
1							
2							
3							
4							

Tabelle 3.1: Firewallregeln aus Sicht von Berlin

3.1.4 Aufgabe 1.4

Nach den oben genannten Anforderungen können Sie leicht zwei der vier Zeilen durch eine einzige Befehlszeile realisieren, so dass Sie insgesamt nur drei Befehlszeilen implementieren müssten. Welche zwei Zeilen sind dies? Denken Sie dabei bereits hergestellte Verbindungen im Rahmen von TCP. Hinweis: Sowohl SSH als auch HTTP basieren auf TCP.

3.1.5 Aufgabe 1.5

Implementieren Sie nun die gewählte default policy sowie die das geplante Regelwerk auf dem Server in Berlin. Notieren Sie dabei alle Befehle.

Geben Sie anschließen Ihre Firewalleinstellungen aus.

Prüfen Sie mit Hilfe der beiden Clients Hamburg und München via http und ssh, ob Ihre Einstellungen erfolgreich waren. Dabei sollen Sie sowohl zeigen, dass die erlaubten Verbindungen erfolgreich hergestellt werden können, unerlaubte jedoch geblockt werden.

Hinweis: Warten Sie beim Testen von ssh einen Fehlversuch, bzw. eine Passwortabfrage ab, denn die Anfragen können unter Umständen ein paar Sekunden dauern! Nutzen Sie von Hamburg aus das Programm links, um sich mit dem Webserver zu verbinden (`links http://<rechner>` oder `wget http://<rechner>`). Eine SSH-Verbindung wird mit `ssh benutzername@IP-Adresse` aufgebaut.

3.1.6 Aufgabe 1.6

Aus Performancegründen kann es sinnvoll sein, die Regel, die Pakete einer bestehenden Verbindung akzeptiert, als erste Regel in eine Kette einzufügen. **Begründen Sie dies.**

3.2 Aufgabe 2: Paket-Weiterleitung

Wieder benutzen wir die filter-Tabelle, diesmal aber zusätzlich mit der Kette FORWARD.

3.2.1 Aufgabe 2.1

Berlin übernimmt die Aufgabe als Router. Begründen Sie diese Entscheidung.

3.2.2 Aufgabe 2.2

Löschen Sie die getätigten Einträge in der filter-Tabelle. Nach dem Löschen soll für alle drei Ketten INPUT, OUTPUT und FORWARD eine „negative default policy“ existieren.

Stellen Sie sicher, dass IP-Forwarding auf Berlin aktiv ist.

3.2.3 Aufgabe 2.3

Nun sollen zwei Arten von Verbindungen ermöglicht werden:

- Der Web-Server in Hamburg soll von München aus erreichbar sein.
- Es soll möglich sein, sich von Hamburg mittels SSH nach München zu verbinden.

Da diese beiden Rechner sich nicht im gleichen Netz befinden, muss der Verkehr in Berlin weitergeleitet werden.

Entwerfen Sie zunächst Firewallregeln mit der unten stehenden Tabelle aus der Sicht von Berlin, welche dies ermöglichen. Benutzen Sie dafür die Kette FORWARD.

Hinweis: Nach der Konfiguration soll die Firewall noch immer nur so wenig wie möglich erlauben.

Nr.	Source	Destination	Protocol	sport	dport	Chain	Action
1							
2							
3							
4							

Tabelle 3.2: Firewallregeln

Beachten Sie, dass es auch hier wieder möglich ist, Regeln zusammenzulegen indem bestehende TCP-Verbindungen akzeptiert werden. Welche Regeln sind dies?

3.2.4 Aufgabe 2.4

Erklären Sie, warum nun die Kette FORWARD benutzt werden muss.

3.2.5 Aufgabe 2.5

Implementieren Sie nun ihr entworfenes Regelwerk mit einer minimalen Anzahl von Befehlen.

Testen nun, ob die gewünschten Verbindungen zustande kommen.

3.3 Aufgabe 3: Network Address Translation

Nun sollen Sie Netz A, bestehend aus Ihrem Arbeitsplatzrechner, Hamburg und Berlin, per Masquerading an Netz B anbinden. Die Routing-Funktion übernimmt Berlin.

3.3.1 Aufgabe 3.1

Setzen Sie zunächst alle Firewalleinstellungen aus den vorherigen Aufgaben zurück.

Setzen Sie nun die default policy der Ketten INPUT und OUTPUT auf DROP. Der Einfachheit halber wird die default policy von FORWARD auf ACCEPT gesetzt, um eine Weiterleitung von Paketen zu ermöglichen.

Vergewissern Sie sich, dass IP-Forwarding aktiviert ist.

Unser Webserver wird nun auf München laufen. Lassen Sie sich mit `tail -f /var/log/apache2/access.log` die letzten Zugriffe auf den Webserver anzeigen. Brechen Sie die Betrachtung nicht ab, sondern lassen Sie sie während der Aufgabe einfach weiterlaufen.

3.3.2 Aufgabe 3.2

Greifen Sie nun via HTTP von Hamburg (z.B. mit links oder wget) und Ihrem Arbeitsplatzrechner (z.B. mit Firefox) auf München zu. Finden Sie den Unterschied zwischen den beiden Clients in der Log-Datei. Wieso existiert der Unterschied?

3.3.3 Aufgabe 3.3

Nun betrachten wir die Tabelle nat. Lassen Sie sich alle Ketten auf dem Router anzeigen und dokumentieren Sie diese. Fügen Sie außerdem eine Regel in der Tabelle POSTROUTING ein, die die Pakete aus dem Subnetz 192.168.23.0/24 nach Subnetz 172.16.9.0/24 maskiert. Beachten Sie dabei welches Interface auf dem Router verwendet wird.

Greifen Sie aus Hamburg auf den Webserver von München zu. Betrachten Sie die Log-Datei und arbeiten Sie die Unterschiede zu den vorherigen Aufgaben des Abschnitts heraus.

3.4 Aufgabe 4: Content Filtering

In der letzten Aufgabe sollen Sie einige Probleme auf dem Rechner in Berlin mit Hilfe von Content Filtering lösen.

3.4.1 Aufgabe 4.1

Setzen Sie zunächst alle Firewalleinstellungen aus den vorherigen Aufgaben zurück. Es sollte eine „positive default policy“ existieren.

Es hat sich herausgestellt, dass Ihr Netzwerk durch den berüchtigten EICAR-Virus [8] kontaminiert wurde. Eine verseuchte Datei (zum Testen) finden Sie unter `http://berlin/file.txt` (Bzw. auf Berlin unter `/var/www/html/file.txt`).

Erklären Sie, was der ËICAR-Virusïst.

3.4.2 Aufgabe 4.2

Verhindern Sie eine weitere Kontaminierung des Netzwerkes, indem Sie mit Hilfe von String-Matching Aufrufe dieser Datei verhindern. Dokumentieren Sie, dass HTTP- und SSH-Verbindungen weiterhin möglich sind. Der Virus darf über HTTP nicht abrufbar sein. Beachten Sie, dass Viren nicht an Dateinamen gebunden sind.

Versuchen Sie, den Virus über HTTP herunter zu laden, etwa mit `links, wget` oder `curl`. Stellen Sie sicher, dass der Download nicht funktioniert. Erklären Sie: Warum ist der Virus über SSH weiterhin abrufbar?

Hinweis: Nutzen Sie Hochkommata anstatt Anführungszeichen in der Shell, damit bestimmte Sonderzeichen nicht interpretiert werden.

3.4.3 Aufgabe 4.3

Das Management hat Ihnen aufgetragen, dass sämtlicher Zugriff auf Seiten mit pornographischen Inhalten gesperrt werden soll.

Geben Sie, wenn möglich, einen iptables-Befehl an, der dieses erreichen kann. Ist dieses eindeutig und einfach machbar? Sie können Ihren Befehl testen, indem Sie die Startseite im Rahmen Ihrer kreativen Möglichkeiten entsprechend modifizieren, dieses ist jedoch optional (Die Startseite befindet sich in `/var/www/html/index.html`).

Ende des Versuchs

3.5 Kontrollfragen

Sie sollen folgende Fragen beantworten können (bevor der Versuch beginnt):

1. Vor welchen Gefahren schützt eine Firewall?
2. Vor welchen Gefahren schützt sie nicht?
3. Ordnen Sie den Schichten des ISO/OSI-Modelles entsprechende Firewallkomponenten zu. Ist dies immer eindeutig möglich? Wenn nein, warum nicht?
4. Warum sollte man möglichst auf eine einstufige Firewallkonfiguration verzichten?
5. Wie könnte eine sinnvolle „default policy“ für den maximalen Schutz aussehen?
6. Die Header welcher Protokolle können durch Router verändert werden? Welche Änderungen hat man zu erwarten?
7. Was ist der Unterschied zwischen TCP und UDP?
8. Was sind die Vorteile von zustandsbehafteten Paketfiltern?
9. Was ist der Unterschied zwischen folgenden Targets: REJECT und DROP?
10. Welchen Zweck erfüllt die Tabelle filter?
11. Erklären Sie die Ketten der filter-Tabelle.
12. Wann trifft die Matching-Option ESTABLISHED zu?
13. Welchen Zweck erfüllt die Tabelle nat?
14. Erklären Sie die Ketten der nat-Tabelle.
15. Wie können Sie als Netzwerkadministrator verhindern, dass Ihre User Filesharing betreiben? Kann man die von Ihnen genannten Möglichkeiten mit iptables realisieren?
16. Überlegen Sie, mit welcher Firewall-Konfiguration Sie den PC dahingehend absichern können, dass Zugriffe auf den HTTP-Port (80) immer möglich sind, während der SSH-Port (22) nur von einem bestimmten Rechner aus erreicht werden kann. Allgemeine Erklärung eines möglichen Befehls reicht.
17. Beschreiben Sie den Weg eines eingehenden Pakets durch die Tabellen filter und nat bis hin zum Serverprogramm.
18. Wenn die Firewall als Router eingesetzt wird, passiert das Paket mehrere Ketten der Tabelle nat und eine filter-Kette. Wo sind welche Einschränkungen / Headermodifikationen möglich? Welche Einflussmöglichkeiten bieten sich damit?
19. Was bedeuten positive Fiterregeln bei Firewalls?
20. Wie kann man einfach alle Matches einer Regelkette negieren?

Literaturverzeichnis

[1] Bundesamt für Sicherheit in der Informationstechnik. IT-Grundschutz-Kataloge. http://www.bsi.bund.de/IT-Grundschutz-Kataloge.

[2] Fuller, Li, Yu & Varadhan. Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy, 1993. http://www.ietf.org/rfc/rfc1519.txt.

[3] Netfilter. http://www.netfilter.org/.

[4] Netfilter. Documentation about the netfilter/iptables project. http://www.netfilter.org/documentation/.

[5] Oskar Andreasson. Iptables Tutorial 1.2.2. http://www.frozentux.net/iptables-tutorial/iptables-tutorial.html.

[6] Tibor Jager. Computernetze. http://www.ei.rub.de/studium/lehrveranstaltungen/616/.

[7] Wikipedia. Denial of Service. http://de.wikipedia.org/wiki/Denial_of_Service.

[8] Wikipedia. EICAR-Testdatei. http://de.wikipedia.org/wiki/EICAR-Testdatei.

[9] Wikipedia. Netzmaske. http://de.wikipedia.org/wiki/Netzmaske.

[10] Wikipedia. Stringmatching. http://de.wikipedia.org/wiki/String-Matching-Algorithmus.