

Ralf Spenneberg

# Linux-Firewalls mit iptables & Co.

Sicherheit mit Kernel 2.4 und 2.6  
für Linux-Server und -Netzwerke



 ADDISON-WESLEY

---

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England  
Don Mills, Ontario • Sydney • Mexico City  
Madrid • Amsterdam



# 4 Bedrohungen bei der Vernetzung von Systemen

Solange die Rechner nicht vernetzt waren, war die Welt noch in Ordnung. Angriffe beschränkten sich auf physikalische Zugänge. Solange die Systeme für Unbefugte unzugänglich aufgestellt wurden, war ihre Sicherheit gewährleistet. Die physikalische Sicherheit des Rechnergehäuses und des Serverraums garantierte auch die Sicherheit der vorhandenen Daten. Sobald jedoch die Rechner eine größere Verbreitung fanden und Massenspeicher in Form von Disketten zum Austausch von Daten eingesetzt wurden, tauchte eine neue Bedrohung in Form von Viren auf. Der erste funktionsfähige Virus wurde von Dr. Fred Cohen in seiner Doktorarbeit 1983 beschrieben<sup>1</sup>. Die tatsächliche Verbreitung begann mit der Verbreitung des IBM PCs und dem zunehmenden Informationsaustausch per Diskette und per Modem.

Durch die Vernetzung haben die Bedrohungen stark zugenommen. Der Angriff ist immer, zu jeder Zeit und von überall fast anonym durchführbar. Um sich für die Verteidigung zu rüsten, ist es wichtig, die Bedrohungen zu kennen und zu verstehen.

## 4.1 Angreifer und Motivation

Zunächst sollten Sie sich einige Gedanken über den potenziellen Angreifer und seine Motivation machen. Hieraus ergeben sich anschließend direkt Schlussfolgerungen zum erforderlichen Aufwand für die Sicherheit.

Die meisten Administratoren denken bei einem Angreifer zunächst an den klassischen Hacker. In den seltensten Fällen ist dieser jedoch für den Angriff verantwortlich.

### 4.1.1 Der klassische Hacker

Der klassische Hacker interessiert sich meist aus Neugierde für die interne Funktion eines Programms oder eines Betriebssystems. Der Hacker unterscheidet sich von dem Cracker, der – mit ähnlichem Wissen ausgestattet – versucht, in fremde

---

<sup>1</sup> Allerdings existierten auch schon früher Programme, die viele Anzeichen eines Virus aufwiesen. Diese Arbeit analysierte aber erstmalig wissenschaftlich das Phänomen.

Systeme einzudringen und dort Schaden anzurichten. Der Hacker verfügt über sehr spezialisiertes Wissen über die verwendeten Programmiersprachen und Algorithmen. Er analysiert die Softwarefunktionen und findet dabei Programmier-, Logik- oder Designfehler. Anschließend versucht er, diese Fehler auszunutzen, um das Programm zu anderen Zwecken zu gebrauchen. Ist er erfolgreich, so hat er einen sogenannten *Exploit* gefunden. Seine Motivation ist meist Neugierde, Wissensdrang, Weiterbildung und »weil es möglich ist«.

Der klassische Hacker ist normalerweise nicht daran interessiert, tatsächlich in fremde Systeme einzudringen. Er veröffentlicht meist die für den Angriff erforderlichen Informationen (teilweise gegen Bezahlung).

Der Chaos Computer Club hat auf seiner Webpage (<http://www.ccc.de/hackerethics>) eine Hacker-Ethik veröffentlicht, mit der sich wahrscheinlich die meisten Hacker identifizieren können.

### 4.1.2 Script-Kiddies

Der Begriff Script-Kiddie bezeichnet Angreifer, die nicht selbst in der Lage sind, eine Sicherheitslücke zu finden, zu analysieren und einen Exploit zu entwickeln. Diese Angreifer sind darauf angewiesen, dass ein anderer Hacker einen Exploit entwickelt, den sie anschließend verwenden. Häufig wird dazu der Exploit in ein Skript eingebunden. Dieses Skript wird anschließend genutzt, um eine Vielzahl von Systemen anzugreifen und dort einzubrechen.

Die Tatsache, dass von Kiddies gesprochen wird, hat keinen Bezug auf das Alter der Personen. Script-Kiddies gibt es in jedem Alter. Es soll eher abfällig zum Ausdruck bringen, dass das Script-Kiddie nicht erfahren genug ist, den Exploit selbst zu entwickeln.

Script-Kiddies werden meist durch Neugierde und die Suche nach Anerkennung in Hacker- und Möchtegern-Hackerkreisen motiviert. Hier versucht ein Script-Kiddie dadurch auf sich aufmerksam zu machen, dass er in besonders viele Systeme eingedrungen ist.

### 4.1.3 Insider

Eine zweite sehr gefährliche Gruppe von potenziellen Angreifern ist der Insider. Hier handelt es sich zum Beispiel um entlassene Mitarbeiter, die über internes Wissen verfügen, oder aber auch um enttäuschte Mitarbeiter, die Rache üben möchten. Des Weiteren kann es sich um Zulieferer, Berater oder andere handeln. Diese Personen sind deshalb so besonders gefährlich, da sie über interne Informationen verfügen, die anderen Angreifern nicht zugänglich sind. Dadurch können sie unter Umständen Sicherheitsvorkehrungen umgehen oder besser angreifen. Meist müssen diese Personen sich keine Gedanken um die Firewall machen, da sie über Zugänge verfügen, die nicht von der Firewall überwacht werden. Dabei kann es sich um den lokalen Zugang im Netz oder um VPN-Zugänge handeln.

Auch wenn diese Gruppe häufig nicht über so hohes technisches Wissen in Bezug auf Sicherheitslücken wie ein Hacker verfügt, ist ihre kriminelle Energie meist höher einzustufen.

### 4.1.4 Mitbewerber

Auch Mitbewerber gehören zu den potenziellen Angreifern. Dabei reichen teilweise sicherlich auch Denial-of-Service-Angriffe aus, um einen wirtschaftlichen Vorteil zu erhalten. Stellen Sie sich vor, Sie suchen einen neuen Partner für das Outsourcing Ihrer E-Mail-Kommunikation inklusive Spam- und Virenschutz. In der heißen Phase der Verhandlung mit zwei potenziellen Partnern ist eines der beiden Unternehmen per E-Mail nicht mehr erreichbar. Welchem der beiden potenziellen Partnern vertrauen Sie mehr? Unternehmen A, das gerade unter einem Denial-of-Service-Angriff (DoS) leidet, oder Unternehmen B, das diesen DoS durchführt oder in Auftrag gegeben hat?

Natürlich sind Ihre Mitbewerber aber auch an Ihren Daten interessiert. Wenn sich also die Gelegenheit ergibt, Zugang zu Ihrer Datenbank zu erhalten, den E-Mail-Verkehr mit Ihren Kunden mitzulesen oder Zugang zu neuen Produktentwicklungen zu erhalten, sind Ihre Mitbewerber sicherlich sehr interessiert.

Ihre Mitbewerber haben meist auch die finanziellen Möglichkeiten, hochqualifizierte Angreifer (Industriespione) zu bezahlen. So können Ihre Mitbewerber selbst dann zur Gefahr werden, wenn sie selbst über kein entsprechendes Wissen verfügen.

### 4.1.5 Geheimdienste

Die Geheimdienste waren schon immer an allen Informationen interessiert. Sie hören die Kommunikation zwischen Regierungen, aber auch zwischen Unternehmen ab. Häufig betreiben sie auch Wirtschaftsspionage mit nationalem Interesse, um Unternehmen aus dem eigenen Land einen Wettbewerbsvorteil zu verschaffen. Die Nachrichtendienste verfügen meist über ausreichend finanzielle Mittel, die erforderlichen Kenntnisse sowie die notwendigen Netzwerke und Computer, um Angriffe und Einbrüche durchzuführen. Sie sind auch in der Lage, schwach verschlüsselte Informationen sehr einfach und schnell zu entschlüsseln.

Schließlich besitzen sie häufig auch die Möglichkeit, sich physikalischen Zugang zu bestimmten Informationen zu verschaffen, wenn ein Computerangriff nicht erfolgreich ist. Häufig ist ein physikalischer Einbruch sogar einfacher.

### 4.1.6 Terroristen und organisierte Kriminalität

Auch Terroristen und das organisierte Verbrechen erkennen im Internet immer mehr ein Potenzial für ihre Geschäfte – sowohl, um eigene Geschäfte über das Internet abzuwickeln, als auch, um durch Angriffe und Einbrüche an sensitive Daten heranzukommen.

So sind die Phishing-Angriffe der Jahre 2004 und 2005 sicherlich noch nicht vorbei. Beim Phishing versuchen organisierte Banden gezielt an Login-Informationen zu gelangen. Auch der Versand von Spam ist immer mehr ein Geschäft von wenigen organisierten Banden. Dabei werden kompromittierte Rechner unwissender Anwender genutzt, um in großer Zahl E-Mails zu versenden, die den Empfänger zum Kauf von Viagra bewegen sollen.

2004 wurde auch bekannt, dass ein Erpresserring versucht hat, Online-Wettbüros mit DoS-Angriffen zu erpressen. Der Erpresserring verfügte angeblich über Kontakte zur russischen Mafia (siehe c't 14/2004 und <http://www.heise.de/ct/04/14/048/default.shtml> und <http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/49292>).

Viele Angriffe sind auch politisch motiviert. So wurden nach dem 11. September 2001 besonders viele Angriffe auf muslimische Websites unternommen.

Derartige Angriffe werden sicherlich in nächster Zukunft stark weiter zunehmen und weitere Medien entdecken. Ich selbst befürchte schon für das Medium VoIP ähnliche Entwicklungen, wie wir sie im Moment bei E-Mail mit Spam erleben. Der Begriff für VoIP-Spam ist bereits gefunden worden. Diese Form wird als SPIT (Spam via Internet-Telefonie) bezeichnet.

## 4.2 Tendenzen und Entwicklungen

Wenn man nun die potenziellen Angreifer, ihr Know-how und ihre kriminelle Energie betrachtet, kann man recht einfach die Grafik 4.1 erzeugen. Diese Grafik zeigt, dass das größte Gefahrenpotenzial von den Geheimdiensten und den Industriespionen ausgeht, die von Mitbewerbern angeheuert werden.

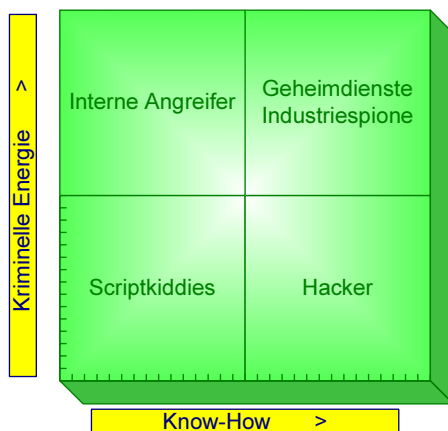


Abbildung 4.1: Das Risiko eines erfolgreichen Angriffs steigt mit steigender krimineller Energie und steigendem Know-how.

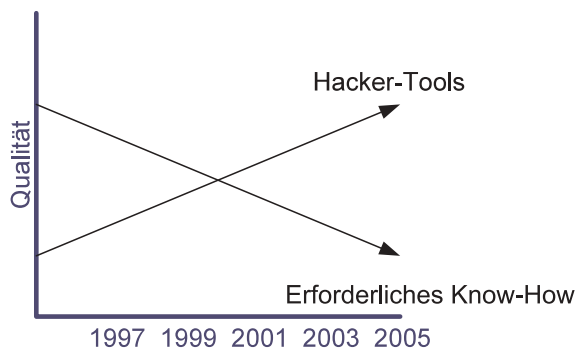


Abbildung 4.2: Die Qualität der Angriffswerkzeuge nimmt seit einigen Jahren stark zu. Das notwendige Know-how sinkt daher stetig.

Allerdings sollten Sie auch die Insider nicht aus den Augen verlieren. Ihre hohe kriminelle Energie macht sie besonders gefährlich. Die Script-Kiddies sind meist zu vernachlässigen. Sie führen keine gezielten Angriffe auf Sie durch, sondern suchen meist allgemein nach verwundbaren Systemen. Sie suchen die niedriger hängenden Früchte, die leicht zu erreichen und anzugreifen sind. Ein gezielter Angriff gegen Ihre Systeme durch Script-Kiddies ist unwahrscheinlich, wenn Sie Ihre Hausaufgaben gemacht haben, Ihre erreichbaren Systeme ordentlich gepatcht sind und Sie über eine Firewall verfügen. Dann zählen Sie nicht zu den niedrig hängenden Früchten. Dennoch müssen Sie möglicherweise mit gezielten Angriffen durch Mitbewerber, Industriespione etc. rechnen. Hier kommt erschwerend hinzu, dass in den letzten Jahren die Angriffswerkzeuge immer besser geworden sind und das erforderliche Know-how des Durchschnittseindringlings entsprechend abgenommen hat (siehe Abbildung 4.2)

### 4.3 Schutzziele

In diesem Abschnitt möchte ich ganz allgemein die zu schützenden Ziele beschreiben. Im Grunde gibt es vier Ziele beim Schutz von Netzen, Rechnern und Daten:

- Integrität
- Vertraulichkeit
- Authentizität
- Verfügbarkeit

Datenintegrität bedeutet, dass eine Änderung der Daten durch Unbefugte nicht unbemerkt bleibt. Die Integrität von Daten wird üblicherweise durch kryptographische Prüfsummen erreicht.

Die Vertraulichkeit der Daten wird meist mit Hilfe von eingeschränkten Leserechten und zusätzlicher Verschlüsselung erreicht. Nur autorisierte Personen erhalten den

Schlüssel für die Entschlüsselung. Das Abhören einer Kommunikation oder das Ausspähen von Daten wird so unmöglich.

Die Authentizität stellt sicher, dass die Daten aus der richtigen Quelle stammen. In einer Kommunikationsbeziehung wird dies durch Authentifizierungen und digitale Signaturen erreicht.

Die Verfügbarkeit von Daten wird häufig vernachlässigt. Dieses Ziel verlangt, dass die gewünschten Daten jederzeit verfügbar sind. Hier werden häufig technische Hilfsmittel, wie Backups, unterbrechungsfreie Stromversorgungen, Cluster etc. genutzt, um die Verfügbarkeit zu sichern.

Sie mögen sich fragen, was diese Ziele nun mit einer Firewall zu tun haben. Eine Firewall ist auch ein Werkzeug, um diese Sicherheitsziele zu erreichen. Sie sichert die Integrität der lokal gespeicherten Daten, indem sie nichtautorisierte Kommunikationsanfragen, zum Beispiel an den Datenbankserver, ablehnt. Sie stellt die Vertraulichkeit von Daten sicher, indem sie verhindert, dass bestimmte Kommunikationsverbindungen von und nach außen aufgebaut werden können. Die Authentizität kann eine Firewall nur begrenzt prüfen. Sie schützt aber vor einfachen Spoofing-Angriffen. Schließlich kann sie auch die Verfügbarkeit der Daten durch die Abwehr eines Denial-of-Service-Angriffs garantieren oder selbst hochverfügbar die Internetanbindung sicherstellen.

In größeren Unternehmen gibt es neben diesen sehr technischen Zielen auch formale Ziele. Hierzu gehören zum Beispiel die Nachvollziehbarkeit der Datenverarbeitung. Jeder Vorgang, zum Beispiel in einer Bank, muss nachvollziehbar sein. Auch die Revisionssicherheit spielt hier häufig eine große Rolle. Die Revisionssicherheit verlangt zusätzliche Protokolle, Erhebungen, Inventuren und spezielle Formen der Datenspeicherung und -haltung. Schließlich spielen auch gesetzliche Rahmenbedingungen eine große Rolle. Das können einfache Gesetze sein, wie zum Beispiel der Datenschutz personenbezogener Daten, aber auch sehr komplexe gesetzliche Rahmenbedingungen, wie Basel-2 und der Sarbanes-Oxley-Act. Da viele Unternehmen heute global arbeiten, unterliegen sie auch international gültigen Gesetzen.

Auch bei der Einhaltung und Erfüllung dieser Rahmenbedingungen kann eine Firewall eine wichtige Aufgabe spielen. Gerade die Protokollierung durch eine Firewall wird hier häufig betroffen sein. Während der Datenschutz des Protokollieren personenbezogener Daten hier kritisch gesehen wird, werden andere Personen in den Protokollen wertvolle Hilfsmittel zur Revisionssicherheit und Sicherstellung der Nachvollziehbarkeit sehen.

Falls Sie nicht eine Firewall für ein Unternehmen der Fortune500 aufbauen wollen, sind diese letzten Überlegungen möglicherweise für Sie uninteressant. Dennoch sollten Sie zum Beispiel den Datenschutz nicht aus dem Blickfeld verlieren. Hier müssen Sie sich bei einer Firewall möglicherweise ausführlich Gedanken machen.

## 4.4 Angriffsmethoden

Die meisten Angriffe lassen sich in wenigen Gruppen kategorisieren. Im Folgenden stelle ich Ihnen die wichtigsten Kategorien vor.

### Hinweis



Ich konzentriere mich im Weiteren auf die sehr technische Seite der möglichen Angriffe über ein Netzwerk. Daneben gibt es natürlich auch noch:

- **Social Engineering.** Hier versucht der Angreifer, durch soziale Kompetenz vertrauliche Informationen, wie zum Beispiel Kennwörter, zu erhalten. Hierzu täuscht er einen legitimen Benutzer, einen Administrator oder andere vertrauenswürdige Personen vor. Kevin Mitnick hat ein sehr eindrucksvolles Buch zu diesem Thema geschrieben (*The Art of Deception*, Kevin Mitnick, 2003).
- **War Dialing.** Anstatt den direkten Zugang durch eine Firewall zu suchen, bemüht sich der Angreifer, alternative Zugänge über unbewachte Modem- oder ISDN-Zugänge zu finden. Hierfür verwendet er Software, die jede erdenkliche Telefonnummer eines Unternehmens anwählt und die Antwort testet.

Zunächst gibt es die große Menge der Denial-of-Service-Angriffe. Hier versucht der Angreifer nicht, die Kontrolle über das System zu erhalten, sondern die Funktion des Systems zu stören. Beim Spoofing täuscht der Angreifer eine falsche Identität als Client oder Server vor. Beim Hijacking versucht ein Angreifer, eine aufgebaute Verbindung zu kapern. Ein Bufferoverflow kann auch einen Denial-of-Service auslösen. Jedoch versucht der Angreifer, wenn möglich mit einem Bufferoverflow die Kontrolle über das System zu übernehmen. Der Bufferoverflow wird durch Programmierfehler in den betroffenen Applikationen möglich. Der Formatstring-Angriff ist dem Bufferoverflow sehr ähnlich und erlaubt den Angreifer ebenfalls häufig die Übernahme der Kontrolle. Bei der Race-Condition handelt es sich um einen Fehler in der Programmlogik. Hierbei konkurrieren zwei Prozesse um eine Ressource. Wenn der Programmierer die Ressource nicht richtig vor dem Zugriff des zweiten Prozesses geschützt hat, kann der Angreifer dies ausnutzen. Die SQL-Injektion ist ein typischer Angriff auf Webserver mit dynamischen datenbankgestützten Webseiten. Hier versucht der Angreifer, die von der Webapplikation verwendeten SQL-Abfragen zu modifizieren. So kann er auf Daten zugreifen, die die Webapplikation normalerweise nicht ausgeben würde, oder sogar die Daten in der Datenbank modifizieren.

Im Folgenden werden die verschiedenen Angriffsmethoden ausführlicher dargestellt. Ich werde Ihnen dann auch verschiedene Möglichkeiten der Abwehr nennen.

### 4.4.1 Denial-of-Service (DoS)

Bei einem Denial-of-Service stört der Angreifer die Funktion eines Dienstes so, dass ein legitimer Nutzer nicht mehr auf diesen Dienst zugreifen kann. Im einfachsten Fall verübt bereits eine Putzfrau, die mit ihrem Staubsauger das Kabel eines Servers aus der Steckdose zieht, bereits unwissentlich einen DoS.

Es gibt viele verschiedene Möglichkeiten, einen DoS durchzuführen. Diese entsprechen den verschiedenen Schichten, auf denen die Kommunikation erfolgt.

- Der Angreifer kann versuchen, die Netzwerkbandbreite des Servers komplett auszulasten, so dass weiterer Verkehr nur sehr langsam transportiert wird. Dieser Angriff erfolgt also auf der physikalischen Schicht und ist durch deren Bandbreite begrenzt. Es ist heute sehr schwierig geworden, diesen Angriff durchzuführen, da die Bandbreiten in den letzten Jahren stark zugenommen haben. Der Angreifer benötigt enorme Ressourcen, um sein Ziel zu erreichen.

Allerdings ist es mit so genannten Bot-Netzen sehr wohl möglich, den nötigen Netzwerkverkehr zu generieren. Bot-Netze sind trojanisierte Rechner (meist Microsoft Windows), die zu Tausenden in einem Netz zusammengeschlossen wurden und gemeinsam von dem Angreifer gesteuert werden. Ein Bot-Netz wird häufig für den Versand von Spam-E-Mails verwendet.

Ein Schutz vor diesem Angriff ist nur mit Unterstützung des Providers möglich, der den Verkehr vielleicht vorher filtern kann. Werden jedoch die Absenderadressen der Pakete gefälscht und diese Pakete von vielen verschiedenen Systemen verschickt, ist das sehr schwer.

- Der nächste mögliche DoS-Angriff greift den TCP-Stack des Servers an und führt einen SYN-Flood durch. Der Angreifer generiert viele TCP-SYN-Pakete. Hierbei handelt es sich um Pakete, die einen Verbindungsaufbau anzeigen. Der Server antwortet auf dieses Paket mit einem SYN/ACK-Paket. Damit sich der Server später an diesen Verbindungsaufbau erinnern kann, speichert er die Verbindungsinformationen in einer Tabelle ab, in der alle schwebenden Verbindungen vorgehalten werden (Pending Connections). Sobald das dritte Paket, das TCP-ACK-Paket des Clients, erhalten wird, erhält die Verbindung den Status einer aufgebauten Verbindung und wird aus der Pending-Connections-Tabelle gelöscht und in der Tabelle der aufgebauten Verbindungen (Established Connections) eingetragen.

Bei einem SYN-Flood überflutet der Angreifer den Server mit einer Vielzahl von TCP-SYN-Paketen. Hierbei fälscht der Angreifer zufällig die Absenderadresse der Pakete. Sinnvoll sind hier Absenderadressen von nicht existenten Rechnern. Der Server wird auf alle TCP-SYN-Pakete mit einem TCP-SYN/ACK-Paket antworten und diese Verbindungen in die Pending-Connections-Tabelle eintragen. Diese Tabelle weist jedoch (ähnlich allen anderen Tabellen in Computern) nur eine begrenzte Größe auf. Sendet der Angreifer mehr Pakete, als Verbindungen in dieser Tabelle gespeichert werden können, so muss der Server Verbindungen aus dieser Tabelle löschen.

Erfolgt gleichzeitig zum SYN-Flood ein korrekter Verbindungsaufbau durch einen echten Client, so kann der Server nicht zwischen dieser Verbindung und den Verbindungen des Angreifers unterscheiden. Sendet der Angreifer die TCP-SYN-Pakete so schnell, dass der Server auch die echte Verbindung aus der Tabelle der Pending Connections entfernen muss, bevor das dritte Paket des TCP-Handshakes vom Client empfangen wird, so wird der Server dieses Paket zurückweisen, da er keine Informationen über die Verbindung mehr besitzt. Die Verbindung wird abgebrochen und der Dienst steht nicht mehr zur Verfügung.

Viele TCP-Implementierungen brechen ab 100 TCP-SYN-Paketen pro Sekunde zusammen. Linux bietet die SYN-Cookies. Diese erlauben immer noch einen Verbindungsaufbau bei einem SYN-Flood mit bis zu 100.000 SYN-Paketen pro Sekunde. Hierbei wird die Tabelle der Pending Connections ignoriert und lediglich auf Basis der Sequenznummer über einen Verbindungsaufbau entschieden. Die Funktionsweise wird in Kapitel 23 beschrieben.

### 4.4.2 Spoofing

Spoofing bezeichnet Angriffe, bei denen der Angreifer bestimmte Informationen fälscht. Meist erfolgt dies, um die Identität eines anderen Rechners anzunehmen. Drei verschiedene Arten des Spoofings werden heute durchgeführt. Hierbei handelt es sich um:

- **IP-Spoofing.** Bei dem IP-Spoofing verändert der Angreifer den IP-Header seiner Pakete. Meist fälscht er seine Absender-IP-Adresse. Hiermit kann er die Identität eines anderen Rechners annehmen. Dies erfolgt, um entweder seine Spuren zu verwischen oder um Vertrauensstellungen zwischen gewissen Rechnern auszunutzen.
- **ARP-Spoofing.** Dies wird besonders von Angreifern in geschwichteten Netzen eingesetzt, um trotz des Einsatzes eines Switches weiterhin sämtliche ausgetauschten Pakete zu protokollieren. Hierbei werden ARP-Antworten gefälscht. Außerdem wird es für ein TCP-Session-Hijacking verwendet (s.u.).
- **DNS-Spoofing.** Hierbei fälscht ein Angreifer die Zuordnung eines DNS-Namens zu einer IP-Adresse, indem er die Antwort eines DNS-Servers fälscht.

#### IP-Spoofing

Dies ist die älteste Variante des Spoofings. Hierbei täuscht der Angreifer eine andere IP-Adresse als Absender vor. Dies wird zum Beispiel beim SYN-Flood verwendet, um die eigenen Spuren zu verwischen.

Das IP-Spoofing kann jedoch auch für einen direkten Angriff genutzt werden. Bei dem SMURF-Angriff sendet der Angreifer ein ICMP-Echo-Request-Paket an die Broadcast-Adresse eines Netzwerks. Sämtliche Unix-Rechner dieses Netzwerks reagieren mit einem ICMP-Echo-Reply-Paket. Der Angreifer erhält hiermit also eine Multiplikation seiner Pakete. Fälscht der Angreifer nun die Absenderadresse so, dass sie die IP-Adresse des anzugreifenden Rechners darstellt, antworten alle Unix-

Rechner bei einem Broadcast-Ping an diesen Rechner. Erfolgt dies häufig genug, so besteht die Möglichkeit, die Netzwerkverbindung des angegriffenen Rechners zu überlasten. Jedoch ist die Antwort auf ein Broadcast-Echo-Request-Paket nur eine Eigenschaft von Unix-Systemen, die heute meist von dem Hersteller oder dem Administrator abgestellt wird.

Schließlich kann das IP-Spoofing auch verwendet werden, um Vertrauensstellungen zwischen Rechnern auszunutzen. Das UDP-Protokoll bietet keinen Schutz vor gespoofen Paketen, da es nicht verbindungsorientiert arbeitet. Bei einer TCP-Verbindung genügt nicht das Fälschen der IP-Adresse. Der Angreifer muss darüber hinaus auch die Sequenz- und Acknowledgement-Nummern spoofen.

Erlaubt zum Beispiel ein Syslogd-Server die Protokollierung von Meldungen mit dem UDP-Protokoll nur bestimmten IP-Adressen, so genügt es in diesem Fall, wenn der Angreifer die IP-Adresse entsprechend fälscht. Das Paket wird dann akzeptiert und die Meldung dementsprechend protokolliert, als käme sie von dem korrekten Rechner.

Ein Schutz vor IP-Spoofing ist nur auf einer Firewall möglich. Hierbei sollten Sie darauf achten, dass die Firewall keine unzulässigen IP-Adressen akzeptiert. Wenn in Ihrer DMZ Adressen aus dem Netzwerk 192.168.0.0/24 verwendet werden, dann dürfen Anfragen aus diesem physikalischen Netz keine andere Absenderadresse verwenden.

### ARP-Spoofing

Beim ARP-Spoofing verfolgt der Angreifer entweder das Ziel, ein TCP-Session-Hijacking durchzuführen (s.u.) oder in einer Umgebung, die durch einen Switch kontrolliert wird, dennoch einen Netzwerksniffer einzusetzen.

Der zweite Angriff soll hier ein wenig ausführlicher betrachtet werden.

Wenn zwei Netzwerkgeräte sich mit dem IP-Protokoll unterhalten möchten, so benötigen sie in einem Ethernet-Netzwerk für die eigentliche Kommunikation zunächst auch noch die Ethernet-MAC-Adressen. Hierfür ist das ARP-Protokoll zuständig. Der Rechner, der ein IP-Paket an die IP-Adresse des Zielsystems senden möchte, sendet zunächst einen ARP-Request, um die dazugehörige MAC-Adresse zu ermitteln. Anschließend sendet er das IP-Paket an die IP-Adresse des Zielsystems und den Ethernet-Rahmen an die MAC-Adresse des Zielsystems.

Wird aus Geschwindigkeitsgründen in diesem Netzwerk ein Switch eingesetzt, so wird dieser zunächst den ARP-Request, da dieser an die Broadcast-Ethernet-Adresse gerichtet ist, an alle angeschlossenen Geräte weiterleiten. Der anschließend versandte Ethernet-Rahmen mit IP-Paket wird jedoch vom Switch nur an das tatsächliche Ziel versandt. Um dies zu ermöglichen, besitzt der Switch eine Liste, in die MAC-Adressen der angeschlossenen Geräte abgespeichert werden.

Bei dem ARP-Spoofing-Angriff (Abbildung 4.3) fälscht der Angreifer (Laptop) den ARP-Reply.

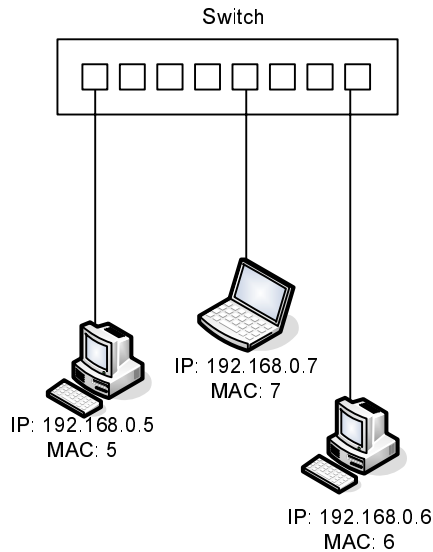


Abbildung 4.3: ARP-Spoofing bei einem Switch.

1. Der Rechner 192.168.0.5 möchte ein Paket an den Rechner 192.168.0.6 senden. Er sendet einen ARP-Request: Welche MAC-Adresse hat der Rechner 192.168.0.6?
2. Der Angreifer beantwortet diese Anfrage nun. Hierzu fälscht er die Antwort, indem er Folgendes sendet: Der Rechner mit der IP-Adresse 192.168.0.6 besitzt die MAC-Adresse 7.
3. Der Rechner 192.168.0.5 sendet nun sein IP-Paket an: Ziel-IP: 192.168.0.6; Ziel-MAC: 7. Der Switch ist nicht in der Lage, die IP-Adressen zu lesen. Er arbeitet lediglich auf der Ebene der MAC-Adressen. Er schlägt in seiner Tabelle nach und stellt fest, dass das Netzwerkgerät mit der MAC-Adresse 7 an Port 4 angeschlossen ist, und leitet das Paket an diesen Port weiter.
4. Der Angreifer muss auf seinem Rechner eine Routing-Software aktivieren. Der Rechner des Angreifers erhält nun das Paket und verarbeitet es, da es an die MAC-Adresse seiner Netzwerkkarte gerichtet ist. Es ist jedoch nicht an seine IP-Adresse gerichtet. Dies ist typisch für einen Router. Wurde das Routing aktiviert, so leitet dieser Rechner das Paket nun an die IP-Adresse 192.168.0.6 und die MAC-Adresse 6 weiter. Als Absender trägt er die IP-Adresse 192.168.0.5 und seine eigene MAC-Adresse 7 ein. Der Switch wird dieses Paket nun beim richtigen Empfänger zustellen. Dieser wird das Paket verarbeiten und umgekehrt antworten.

Es existieren fertige Werkzeuge, die in der Lage sind, unter Linux ein ARP-Spoofing durchzuführen. Ein Schutz vor ARP-Spoofing ist nur durch die folgenden Maßnahmen erreichbar:

- **Statische ARP-Tabellen.** Die Pflege dieser Tabellen ist aber sehr aufwendig. Sie können unter Linux einen statischen Eintrag mit dem Befehl `arp -s "ip" "mac"` erzeugen.
- **ARPwatch.** Der ARPwatch-Daemon ist in den meisten Linux-Distributionen enthalten und ist eine Art ARP-Intrusion-Detection-System. Er überwacht alle ARP-Pakete und meldet Änderungen und damit auch Angriffe.
- **Port-Security.** Viele professionelle Switches sind verwaltbar und bieten die Möglichkeit, die Anzahl der an einem Port erlaubten MAC-Adressen zu beschränken. Bei Überschreitung schaltet der Switch den Port ab.

## DNS-Spoofing

Das DNS-Spoofing führt einen Angriff auf DNS-Ebene durch, der dem Angriff auf ARP-Ebene gleicht. Ein Beispiel wird in der Abbildung 4.4 dargestellt.

Beim DNS-Spoofing sind die folgenden Schritte erforderlich:

1. Der Client 192.168.0.5 möchte eine Netzwerkverbindung mit dem Rechner *www.sparkasse.de* aufbauen. Hierzu benötigt er zunächst die IP-Adresse dieses Rechners. Um diese zu ermitteln, kontaktiert er seinen DNS-Server und fragt ihn nach der IP-Adresse von *www.sparkasse.de*.
2. Da für diese DNS-Anfrage üblicherweise zunächst das UDP-Protokoll eingesetzt wird, existiert kein Schutz gegen gespoofte Pakete. Der Angreifer antwortet an der Stelle des echten DNS-Servers (schneller als der echte DNS-Server) und teilt dem Client mit, dass der Rechner *www.sparkasse.de* die IP-Adresse 192.168.0.7 aufweist.

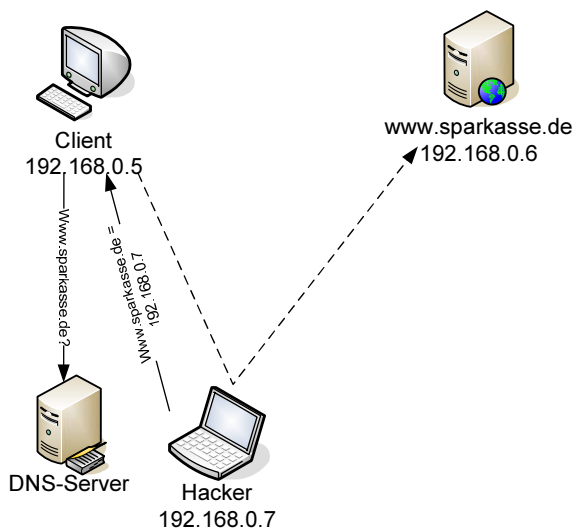


Abbildung 4.4: DNS-Spoofing

3. Der Client wird sich nun mit diesem Rechner verbinden und in Wirklichkeit den Rechner *www.sparkasse.de* erwarten. Damit der Client diese Illusion erhält, richtet der Angreifer auf dem Rechner einen Proxy ein, der sämtliche Anfragen an den echten Rechner weiterleitet und die Antworten an den Client übermittelt.
4. Dieser Angriff ist leicht durch die Verwendung von SSL zu vereiteln. Dazu ist aber zusätzlich erforderlich, dass der Client das SSL-Protokoll auch korrekt einsetzt und die Authentifizierung des Servers überprüft.

Es existieren fertige Werkzeuge, die in der Lage sind, unter Linux ein DNS-Spoofing durchzuführen. Auch existieren Werkzeuge, die dann als SSL-Proxy versuchen, eine SSL-Verbindung zu unterwandern. Wenn der Client die SSL-Zertifikate nicht richtig prüft, kann ein Angreifer so eine Verbindung abhören.

### Gespoofter Portscan

Normalerweise ist ein gespoofter Portscan nicht möglich. Bei einem Portscan versucht der Angreifer, Daten über das untersuchte System zu ermitteln. Spooft er jedoch seine Absenderadresse, so erhält er nie das Ergebnis seines Scans.

Verschiedene Werkzeuge wie zum Beispiel Nmap (<http://www.nmap.org>) versuchen das Problem zu lösen, indem sie jedes Paket in einem Scan mehrfach senden. Alle zusätzlichen Pakete weisen eine gefälschte Absenderadresse (so genannte Decoys) auf. Dies erschwert eine Analyse und Bestimmung des Ursprungs des Portscans sehr oder macht diese Bestimmung sogar unmöglich.

Jedoch gibt es auch die Möglichkeit, einen echten gespooften Portscan durchzuführen. Hierzu ist ein dritter Rechner erforderlich. Dieser Rechner sollte gleichzeitig keine aktiven Netzwerkverbindungen besitzen. Daher wird er als *Silent Host* bezeichnet. Abbildung 4.5 demonstriert die Vorgehensweise.

Der Angreifer sucht zunächst einen so genannten *Silent Host*. Dies ist ein Rechner, der gleichzeitig keine anderen Netzwerkverbindungen unterhält. Hierbei kann es

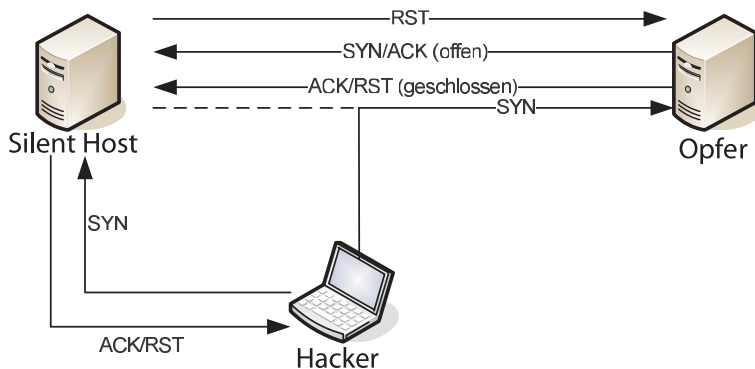


Abbildung 4.5: Gespoofter Portscan

sich zum Beispiel um einen Windows 98-Rechner einer asiatischen Universität handeln. Er beginnt nun, TCP-SYN-Pakete in regelmäßigen Abständen (1/Sekunde) an einen geschlossenen Port zu senden. Er erhält für jedes Paket ein TCP-RST/ACK-Paket zurück. Diese Pakete weisen eine steigende IP-Identifikationsnummer auf (Abbildung C.2 auf Seite 601). Wenn der Rechner gleichzeitig keine weiteren Pakete versendet, so wird diese Nummer immer um 1 inkrementiert. (Achtung: Windows vertauscht die beiden Bytes der Identifikationsnummer. Unter Linux erscheint damit der Inkrementierungsschritt als 256.)

Nun beginnt der Angreifer mit dem Portscan. Er sendet mehrere TCP-SYN-Pakete in den gleichen regelmäßigen Abständen an das Opfer. Hierbei fälscht er die Absenderadresse so, dass das Opfer seine Antworten an den *Silent Host* sendet. Es existieren nun zwei Möglichkeiten: Der Port ist offen, oder der Port ist geschlossen.

1. **Offen:** Wenn der Port auf dem Opfer offen ist, so antwortet das Opfer mit einem TCP-SYN/ACK-Paket an den *Silent Host*. Dieser kann dieses Paket nicht zuordnen und sendet ein TCP-RST-Paket an das Opfer. Damit ist die Verbindung für alle beendet.
2. **Geschlossen:** Wenn der Port auf dem Opfer geschlossen ist, so antwortet das Opfer mit einem TCP-RST/ACK-Paket an den *Silent Host*. Dieser reagiert auf das Paket nicht mit einem weiteren Paket.

Da der *Silent Host*, wenn der Port auf dem Opfer offen war, nun weitere Pakete in regelmäßigen Abständen an das Opfer versendet, wird die IP-Identifikationsnummer der Pakete, die vom *Silent Host* an den Angreifer gesendet werden, immer um 2 inkrementiert. Dies ist ein Zeichen, dass der Port offen war. Ändert sich dies nicht, so war der Port geschlossen.

Das Opfer vermutet, dass es vom *Silent Host* gescannt wird. Wenn der *Silent Host* nicht durch eine Firewall überwacht wird, so kann die Herkunft des Portscans nicht festgestellt werden.

Nmap kann diesen Scan seit einigen Versionen auch durchführen. Hier heißt der Scan »Idle-Scan« und der Silent-Host wird als »Zombie« bezeichnet. Sie benötigen also nicht unbedingt `hping`, wenn Sie bereits Nmap installiert haben.

Vor diesem Scan können Sie sich nicht schützen. Sie sollten lediglich bei der Analyse der Protokolle Ihrer Firewall darauf achten, dass fast alle Informationen in den Protokollen falsch sein können.

### 4.4.3 Session Hijacking

Der Mitnick-Angriff hat die Verwundbarkeit des TCP-Protokolls vorgeführt, wenn die initialen Sequenznummern vorhergesagt werden können. Aber selbst wenn das nicht der Fall ist, ist ein Übernehmen der Verbindung durch einen Angreifer (Session Hijacking) möglich. Hierzu ist es jedoch erforderlich, dass der Angreifer in

der Lage ist, die Verbindung zu beobachten. Er kann dann die verwendeten TCP-Sequenznummern direkt von den ausgetauschten Paketen ablesen.

Durch eine sinnvolle Konstruktion von gespoofen Paketen ist es dann möglich, eigene Daten in diese Verbindung zu injizieren. Dieser Angriff war in der Vergangenheit sehr kompliziert und schwer durchzuführen. Seit einigen Jahren existieren jedoch mehrere Werkzeuge, die dies stark vereinfachen. Die bekanntesten Werkzeuge sind `juggernaut` und `hunt`.

Im Folgenden soll schematisch die Funktionsweise von `hunt` erläutert werden.

1. Wenn `hunt` eine laufende Verbindung beobachtet, so ist es in der Lage, diese Verbindung zu übernehmen. Das Programm `hunt` ist hierbei für Rlogin- und Telnet-Verbindungen optimiert. Der Angriff erfolgt, in dem `hunt` sowohl den Client als auch den Server davor überzeugt, dass sie die Pakete an andere MAC-Adressen versenden müssen (ARP-Spoofing). Anschließend senden sowohl Client als auch Server weiterhin korrekte TCP/IP-Pakete. Diese werden jedoch nicht mehr von der Gegenseite gesehen, da sie an falsche und unter Umständen nicht existente MAC-Adressen gerichtet sind. `hunt` sieht jedoch weiterhin diese Pakete und ist in der Lage, die wichtigen Informationen zwischen Client und Server auszutauschen.
2. Nun kann `hunt` weitere Informationen in den Fluss dieser Verbindung injizieren. Da `hunt` die TCP-Pakete sieht, kann `hunt` die erforderlichen Sequenznummern berechnen.
3. Der Server wird den Empfang der zusätzlichen Daten bestätigen. Da er jedoch das ACK-Paket an die korrekte IP-Adresse des Clients sendet, aber die falsche MAC-Adresse nutzt, sieht der Client diese Bestätigung nicht und sendet keine Fehlermeldung an den Server. `hunt` ist weiterhin in der Lage, weitere Daten mit der Vertrauensstellung des Clients an den Server zu senden.
4. Nachdem die Injektion erfolgt ist, bestehen zwei grundsätzliche Möglichkeiten. `hunt` ist in der Lage, die Verbindung mit einem RST-Paket abubrechen. `hunt` ist aber auch in der Lage, die Verbindung zu resynchronisieren. Hierbei werden einige Daten an den Client und den Server gesendet. Außerdem ist es erforderlich, dass der Benutzer auf dem Client weitere Daten eingibt. Dann kann eine Resynchronisation erfolgreich sein. Der Client hat den Eindruck, dass die Netzwerkverbindung lediglich vorübergehend ausgefallen ist.

Das Werkzeug `hunt` ist auf der Homepage von Pavel Krauz erhältlich (<http://lin.fsid.cvut.cz/~kra/index.html>). Leider funktioniert diese URL aktuell nicht. Sie können das Werkzeug aber alternativ von <http://packetstorm.linuxsecurity.com/sniffers/hunt/> herunterladen.

Die TCP/IP-Protokolle bieten keinen Schutz vor diesen Angriffen. Ein Schutz ist hier nur durch eine zusätzliche Signatur der Pakete, durch höhere Protokolle oder

ein VPN möglich. Leider bietet TCP/IP keine Möglichkeit, die Authentizität eines Pakets zu überprüfen. Viele Applikationen wie Telnet führen die Authentifizierung aber nur zu Beginn durch. Anschließend gilt die Verbindung als authentifiziert. Ein Angreifer kann die Verbindung übernehmen und ist dann ebenfalls authentifiziert.

#### 4.4.4 Bufferoverflow

Ein Bufferoverflow ist das Ergebnis eines Programmierfehlers. Nicht jeder Programmierfehler führt zu einem Bufferoverflow, und nicht jeder mögliche Bufferoverflow kann von einem Angreifer ausgenutzt werden. Um den Bufferoverflow zu verstehen, müssen Sie sich in die Rolle eines Programmierers hineinversetzen.

In vielen Programmen kommen einzelne Aufgaben wiederkehrend vor. Diese Aufgaben werden dann gern in einem Unterprogramm realisiert, das von verschiedenen Stellen des Hauptprogramms aufgerufen werden kann. Damit das Unterprogramm später weiß, wohin es im Hauptprogramm zurückspringen muss, sichert der Prozessor vor dem Aufruf des Unterprogramms den aktuellen Stand des Befehlszeigers (Instruction Pointer, IP) auf dem Stapel (Stack). Der Stapel ist eine dynamische Struktur, auf der ein Programm vorübergehend Daten ablegen kann. Jedes Programm besitzt einen eigenen Stapel. Ein Stapel erlaubt lediglich das Lesen und Schreiben der Daten auf dem höchsten Punkt. Dieser wird mit dem Stapelzeiger (Stackpointer) referenziert (siehe Abbildung 4.6). Benötigt nun das Unterprogramm vorübergehend Speicherplatz für eine Variable, so fordert es diesen Puffer (Buffer) auf dem Stapel an. Der Stapelzeiger wird um die entsprechende Anzahl Bytes verschoben und als Referenz an das Unterprogramm zurückgegeben (siehe Abbildung 4.7). Das bedeutet, das Unterprogramm kann nun Daten auf dem Stapel ablegen, wobei es beim Stapelzeiger beginnt und dann rückwärts geht.

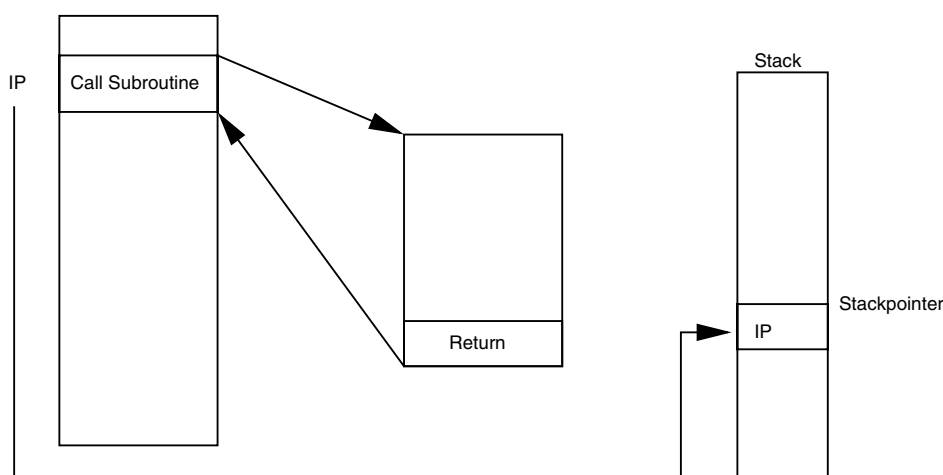


Abbildung 4.6: Funktionsweise eines Bufferoverflows

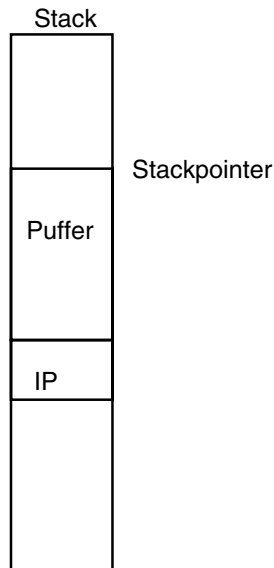


Abbildung 4.7: Pufferreservierung auf dem Stapel

Stellen wir uns vor, das Programm erwartet die Eingabe des Geburtsdatums des Benutzers. So genügen sicherlich 32 Bytes, um diese Eingabe entgegenzunehmen. Diese 32 Bytes werden nun auf dem Stapel reserviert. Aus irgendeinem Grund (Unwissenheit, Börsartigkeit) gibt der Benutzer jedoch 100 Zeichen ein. Überprüft der Programmierer vor der Kopie der Zeichenkette in den dynamisch allozierten Datenbereich auf dem Stapel ihre Länge nicht, so werden alle 100 Bytes auf den Stapel kopiert. Dadurch werden auch Bereiche überschrieben, die andere gültige Daten enthielten. Es kommt zu einem Überlaufen (Overflow) des originalen Puffers (siehe Abbildung 4.8). Hierbei können auch zum Beispiel Rücksprungadressen der Unterprogramme überschrieben werden.

Es gibt einige Programmiersprachen, die dem Programmierer diese Arbeit (das so genannte Boundary Checking) abnehmen, jedoch gehören die Programmiersprachen Assembler und C nicht dazu. In C werden die meisten Anwendungen für Unix- und auch für Windows-Betriebssysteme programmiert.

Handelt es sich um willkürliche Daten, so stürzt das Programm ab und es kommt zum Segmentation Fault (Linux) oder zu einer allgemeinen Schutzverletzung (General Protection Fault, Windows). Der Prozessor erkennt, dass es sich um eine unerlaubte Rücksprungadresse handelt. Ein Zugriff auf den Speicher an der Zieladresse ist dem Programm nicht erlaubt. Das Programm wird von dem Prozessor beendet (stürzt ab) und steht nicht mehr zur Verfügung. Dies ist ein Denial-of-Service.

Unter Umständen besteht jedoch auch die Möglichkeit, den Ort einer Rücksprungadresse auf dem Stack vorherzusagen und so zu überschreiben, dass ein gezielt-

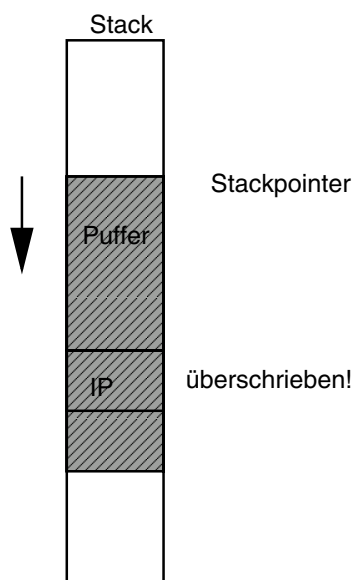


Abbildung 4.8: Überlaufen des Puffers

ter Rücksprung auf Code erfolgt, der sich im Vorfeld in den eingegebenen Daten befand. Dann wird in dem Benutzerkontext des missbrauchten Programms der gerade eingegebene Code ausgeführt. Üblicherweise handelt es sich hierbei um den Aufruf einer Unix-Shell. Daher bezeichnet man diesen von dem Einbrecher verwendeten Code auch häufig als Shell-Code. Dies ist unter Unix besonders brisant, da viele Netzwerkdienste über Rootprivilegien verfügen und Eingaben aus ungewisser Quelle entgegennehmen. Weisen diese Dienste derartige Mängel auf, so können sie ausgenutzt werden, um Rootprivilegien auf dem entsprechenden Rechner zu erlangen. In diesem Fall spricht man von Root Exploits.

Viele der ersten geschriebenen Bufferoverflows sind leicht an dem so genannten NOP Sled zu erkennen. Ein gezielter Rücksprung, wie gerade beschrieben, ist meist nicht möglich. Der Angreifer kann nur selten genau den Zustand des Stacks vorher-sagen. Daher kann er auch nicht die Rücksprungadresse berechnen. Nun versieht er seinen Code zu Beginn mit bis zu mehreren hundert NOP-Befehlen. Ein NOP ist ein No-Operation-Befehl. Dieser Befehl hat, wenn er von dem Prozessor ausgeführt wird, keine Funktion, außer den Instruction-Pointer weiterzubewegen. Ist der Code für den Bufferoverflow derartig angepasst, so muss die Rücksprungadresse nur noch ungefähr in den Bereich der NOP-Befehle zeigen. Dies bezeichnet man als NOP-Schlitten (Sled), da der Befehlszeiger wie auf einem Schlitten über die NOPs zum Bufferoverflow rutscht und schließlich diesen Code ausführt.

Der injizierte Code wird als Shell-Code bezeichnet. Dieser Name leitet sich aus der Tatsache ab, dass klassische Bufferoverflows versuchen, eine Shell zu starten und so Kommandozeilenzugriff auf dem angegriffenen System zu erhalten.

Es gibt verschiedene Möglichkeiten, sich vor Bufferoverflows zu schützen. Der älteste Schutz erreicht dies, indem der gesamte Stapel für das Programm als nicht-ausführbar gekennzeichnet wird.



### Achtung

Machen Sie nicht den Fehler anzunehmen, dass Sie mit dieser Methode aus dem Schneider sind. Erstens stürzt das Programm weiterhin ab. Der Prozessor erkennt lediglich, dass unerlaubter Code angesprungen werden soll. Außerdem gibt es auch Heap-Overflows. Hier hilft die Methode nicht.

Bei älteren Prozessoren der i386-Architektur ist dies nur mit Tricks möglich. Aktuelle Prozessoren unterstützen diese Funktion in Hardware. Bei dem Athlon64 existiert das No-eXecute-Bit (NX), das von AMD als Enhanced-Virus-Protection Technologie vermarktet wird. Intel hat bei den Itanium- und den neuesten Pentium 4- und M-Modellen das eXecute-Disable-Bit (XD) eingeführt. PowerPC, SPARC und Alpha-Prozessoren verfügen über diese Funktion schon länger.



### Achtung

Wichtig zu wissen beim NX- und XD-Bit ist, dass diese Bits nicht automatisch aktiviert werden. Sie müssen ein Betriebssystem einsetzen, das diese Bits auch nutzt (zum Beispiel RHEL4 und Fedora Core mit ExecShield).

Es existieren noch drei weitere Varianten, um sich vor den Gefahren des Bufferoverflows zu schützen. Der modifizierte GNU-C-Compiler Stackguard implementierte als Erster den Schutz mit dem so genannten Canary (Kanarienvogel). Hierbei erzeugt der Compiler modifizierten Code, der vor jedem Sprung in ein Unterprogramm eine Zufallszahl auf dem Stapel abspeichert. Vor jedem Rücksprung überprüft das Programm, ob die Zufallszahl modifiziert wurde. Ist dies der Fall, wird ein Bufferoverflow angenommen und die Ausführung abgebrochen. Ansonsten fährt das Programm fort. Diese Vorgehensweise ist von IBM in ProPolice weiterentwickelt worden. ProPolice gibt es für einige Linux-Distributionen und OpenBSD.

Red Hat hat mit den Position-Independent-Executables (PIE) einen Schritt in eine andere Richtung gemacht. Auch hier wurde der GNU-C-Compiler so modifiziert, dass der entstehende Programmcode beliebig im Speicher arrangiert werden kann. Ein modifizierter Kernel (ExecShield-Patch) kann nun bei jedem Aufruf das Programm und alle Daten inklusive des Stapels zufällig an einer anderen Position abspeichern. Der Angreifer hat nun das Problem, dass er die Rücksprungadressen nicht mehr vorhersagen kann. Stellen Sie sich vor, Sie wären in einem Raum mit

fünfhundert Türen. Eine ist offen. Sie haben nur einen Versuch. Es ist einfach die richtige Tür zu finden, wenn sie immer an derselben Stelle ist und Sie dies vorher in einem anderen Raum üben können. Wenn die richtige Tür aber jedes Mal eine andere ist, ist dies vielfach schwerer.

Ihre letzte Schutzmöglichkeit vor Bufferoverflows ist der Austausch zentraler C-Funktionen. Die meisten Bufferoverflows nutzen spezielle C-Funktionen aus (z.B. `strcpy`). Die Libsafe-Bibliothek bietet Ihnen die Möglichkeit, diese Funktionen gegen sichere Varianten auszutauschen. Die neuen C-Funktionen überprüfen vor jedem Aufruf die Größe des zur Verfügung stehenden Puffers. Libsafe können Sie auf <http://www.research.avayalabs.com/project/libsafe/> herunterladen. Da Libsafe in Kombination mit dem Prelude-IDS (<http://www.prelude-ids.org>) als Intrusion-Detection-System eingesetzt werden kann, finden Sie in meinem Buch »Intrusion Detection and Prevention mit Snort 2 & Co.« weitere Hinweise zur Installation.



#### Achtung

Eine Firewall kann nicht vor einem Bufferoverflow schützen.

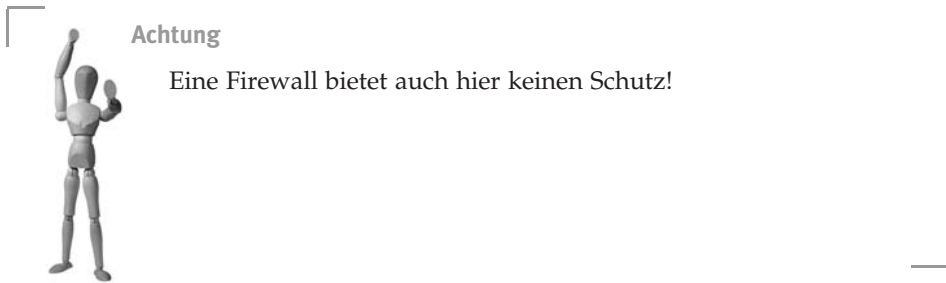
### 4.4.5 Formatstring-Angriffe

Bei den Formatstring-Angriffen handelt es sich um eine noch recht junge Gruppe von Angriffen. Der Formatstring-Angriff wurde zum ersten Mal im Juni 2000 für einen Angriff auf den WU-Ftpd-Server verwendet. Ähnlich wie der Bufferoverflow betrifft dieses Problem Programme, die in C geschrieben wurden. Die Programmiersprache C kennt verschiedene Funktionen, die formatierte Textausgaben erlauben: `printf`, `scanf` etc. Diese Funktionen nehmen einen Formatstring und mehrere Werte entgegen und geben die Werte formatiert aus.

```
printf("%s", buffer)
```

Leider geben nicht alle Programmierer immer den Formatstring an, sondern sparen sich die Mühe und schreiben `printf(buffer)`. Wenn nun der Inhalt des Puffers von dem Angreifer eingegeben wurde, kann er auch Formatstrings angeben. Die Formatstrings und der Puffer werden immer über den Stapel an die Funktion übergeben. Die `printf`-Funktion interpretiert dann die Formatstrings des Angreifers. Es gibt Formatstrings, mit denen der Angreifer sowohl Daten von dem Stapel lesen kann (zum Beispiel `%s` und `%x`) als auch Werte schreiben kann (`%n`). So kann der Angreifer zunächst den Ort der richtigen Rücksprungadresse eines Unterprogramms ermitteln und anschließend überschreiben. Die Auswirkung ähnelt also einem Bufferoverflow. Ein Formatstring-Angriff kann also auch verwendet werden, um die Kontrolle über einen Dienst oder ein System zu erlangen. Dabei besteht mit dem

Formatstring %n die Möglichkeit, an beliebigen Adressen den Inhalt zu modifizieren. Der so von dem Angreifer injizierte Code muss also nicht auf dem Stapel liegen. Die Schutzmechanismen, die einen Überlauf erkennen (Stackguard, ProPolice) oder den Stapel als nicht-ausführbar deklarieren, können uns daher nicht vor Formatstring-Angriffen wirksam schützen. Lediglich die Randomisierung der Adressen (ExecShield mit PIE, PaX etc.) bietet hier einen gewissen Schutz.



### 4.4.6 Race-Condition

Eine Race-Condition entsteht, wenn zwei Prozesse gleichzeitig auf eine Ressource zugreifen, dieser Zugriff nicht geordnet erfolgt und der Ausgang von der Reihenfolge des Zugriffs abhängig ist. Diesen langen komplizierten Satz möchte ich anhand eines einfachen Beispiels erläutern.

Stellen Sie sich vor, Sie haben bei einer Bank ein Konto. Das Konto weist ein Guthaben von 500,00 EUR auf. Sie tätigen eine Überweisung und spenden 250,00 EUR für die Free-Software-Foundation-Europe (FSFE). Gleichzeitig geht Ihr Gehalt von Ihrem Arbeitgeber ein (2.500,00 EUR). Wenn diese beiden Vorgänge nun nicht geordnet durchgeführt werden, kann Folgendes passieren:

1. Für die Spendenüberweisung wird das Guthaben gelesen. Guthaben = 500,00 EUR.
2. Für die Gehaltsüberweisung wird das Guthaben gelesen. Guthaben = 500,00 EUR.
3. Das Gehalt wird addiert, und die Summe wird als neues Guthaben geführt. Guthaben = 500,00 EUR + 2.500,00 EUR = 3.000 EUR.

Diese Summe wird als Guthaben gespeichert.

4. Für die Überweisung wird von dem Guthaben (500,00 EUR, es wurde ja früher gelesen) 250,00 EUR abgezogen. Guthaben = 500,00 EUR - 250,00 EUR = 250,00 EUR.

Diese Summe wird nun als neues Guthaben gespeichert. Die 2.500,00 EUR sind verloren.

Natürlich hätte es auch genau andersherum passieren können. Dann hätten Sie trotz der Überweisung von 250,00 EUR an die FSFE anschließend immer noch 3.000,00 EUR besessen.

Eine Race-Condition führt also zu inkonsistenten Daten. Teilweise kann ein Angreifer diese aber auch ausnutzen, um erweiterte Rechte zu erlangen. Meist erfordert das Ausnutzen einer Race-Condition bereits Zugang zu einem System. Race-Conditions, die aus der Ferne über Netzwerkdienste ausgenutzt werden können, sind sehr selten.

Im Folgenden möchte ich Ihnen zwei typische Race-Conditions vorstellen: /tmp-Race-Conditions und die ptrace-Race-Condition.

### Race-Conditions bei der Behandlung von temporären Dateien

Häufig treten bei der Behandlung von temporären Dateien Race-Conditions auf. Dies hängt damit zusammen, dass bei einem mehrfachen Zugriff auf temporäre Dateien über den Dateinamen das Programm nicht garantieren kann, dass es sich immer um dieselbe Datei handelt.

Stellen Sie sich vor, dass Sie ein Skript entwickeln möchten, das bei allen Systemkonten die Shell auf /bin/false ändern sollte. Ein Systemkonto benötigt keine Möglichkeit der interaktiven Anmeldung auf einem Linux-System. Ihr Skript könnte folgendermaßen aussehen:

#### Listing 4.1: Ein per Race-Condition verwundbares Skript

```
#!/bin/sh

TEMP=/tmp/mytemp

rm -f $TEMP

cat /etc/passwd | while read zeile
do
    echo $zeile | awk -F':' \
        '{if ((>0) && (<500)) print $1":"$2":"$3":"$4":"$5":"$6":/bin/false";
         else print $0}'
done >> $TEMP

mv $TEMP /etc/passwd
chmod 644 /etc/passwd
```

Dieses Skript löscht zunächst die temporäre Datei, da es anschließend eine Zeile an die neue Datei anhängen möchte (`done >> $TEMP`). Dann liest es die Datei `/etc/passwd` und ersetzt bei allen Konten mit den Nummern 1-499 die Shell in der Spalte 7 durch

`/bin/false`. Anschließend benennt das Skript die Datei in `/etc/passwd` um und setzt die Rechte entsprechend.

Wo ist hier die Race-Condition? Stellen Sie sich vor, der Angreifer wüsste, dass Sie dieses Skript als `root` starten. Er könnte versuchen, ein eigenes Skript gleichzeitig laufen zu lassen.

### Listing 4.2: Der Exploit

```
#!/bin/sh

TEMP=/tmp/mytemp

touch $TEMP

while test -f $TEMP
do
  NOP=0 # do nothing
done
echo "toor::0:0:toor owns your machine:./bin/sh" > $TEMP
```

Dieses Skript legt die temporäre Datei mit dem bekannten Namen an. Anschließend prüft das Skript, ob die Datei noch existiert oder bereits von dem verwundbaren Skript gelöscht wurde. Sobald das erkannt wurde, bricht die Schleife ab und das Skript schreibt die angegebene Zeile in die Datei `/tmp/mytemp`, bevor das verwundbare Programm seine Daten an diese Datei anhängt.

#### Tipp



Wenn Sie dieses Problem nachstellen möchten, dann sollten Sie, damit es nicht von Ihrem Glück abhängt, in dem verwundbaren Programm die Zeitspanne des möglichen Exploits vergrößern. Fügen Sie einfach ein `sleep 1` nach dem `rm`-Befehl ein.

Um sich vor diesen Race-Conditions bei der Verwendung von temporären Dateien zu schützen, ist es wichtig, dass der Name der temporären Datei möglichst nicht vorhersagbar ist. Viele Programme hängen daher ihre PID (Prozess-ID) an den Dateinamen an. Jedoch ist auch diese PID auf vielen Linux/Unix-Systemen in gewissen Bereichen vorhersagbar. Die beste Lösung ist die Verwendung von benutzer-spezifischen temporären Verzeichnissen, in denen andere Benutzer keine Dateien erzeugen können, oder der Befehl `mktemp`, der eine temporäre Datei nach dem Muster `/tmp/tmp.XXXXXXXXXX` erzeugt. Hiermit sind  $26^{10}$  verschiedene Dateinamen möglich. Eine Vorhersage durch den Angreifer ist unmöglich.



### Achtung

Bei der Beschreibung des Angriffs sollte Ihnen deutlich geworden sein, dass eine Firewall keinerlei Schutz bietet. Hier kann aber ein MAC-System wie SELinux durchaus Abhilfe schaffen.

### Die ptrace-Race-Condition

Hierbei handelt es sich um eine Race-Condition im Linux-Kernel, die im Januar 2003 entdeckt wurde. Diese Sicherheitslücke war nur lokal ausnutzbar. Dennoch wurde sie in vielen Angriffen genutzt, denn es gab gleichzeitig auch Sicherheitslücken in weiteren Netzwerkdiensten wie dem Apache-Webserver. Die Angreifer konnten über den Netzwerkdienst Kommandozeilenzugriff auf das Zielsystem erhalten. Da die meisten Netzwerkdienste aber nicht über root-Privilegien verfügen, musste der Angreifer anschließend nach einer weiteren Sicherheitslücke suchen, um eine Privilege-Escalation, eine Erweiterung seiner Privilegien, durchzuführen. Dafür wurde dann die ptrace-Race-Condition genutzt. Linux-Kernel < 2.2.25/2.4.20 weisen diese Sicherheitslücke auf.

Der Fehler tritt auf, wenn ein Prozess eine Funktion nutzen möchte, die über ein noch nicht geladenes Kernelmodul realisiert wird. Dies ist zum Beispiel häufig beim Öffnen eines Netzwerk-Sockets für eine untypische Protokoll-Familie (PF\_IPX, PF\_X25, PF\_AX25, PF\_APPLETALK etc.) der Fall. Um die Protokoll-Familie zu unterstützen, muss der Kernel ein Modul nachladen. Hierzu erzeugt der Kernel automatisch von dem anfordernden Prozess einen Kindprozess. Anschließend stellt der Kernel die User-ID des Kindprozesses auf 0 und lädt mit dem Befehl `modprobe` das Modul. Da der Kindprozess in dem Kontext des ursprünglichen Benutzers gestartet wird, kann dieser auf den Prozess Einfluss nehmen, bevor der Kernel die root-Privilegien überträgt. Hier ist die Race-Condition.

Wie nutzt man dies nun aus? Der Kernel bietet mit der ptrace-Funktion die Möglichkeit, Prozesse zu beobachten, Breakpoints einzufügen und den Code zu modifizieren. Diese Funktion wird von Debuggern zur Fehlersuche verwendet. Kann der Angreifer mit dieser Funktion sich an den Kindprozess binden, bevor der Kernel die root-Privilegien überträgt, so kann er anschließend einen Prozess mit root-Privilegien steuern, den Code modifizieren und so jede beliebige Tätigkeit auf dem System ausüben. Er hat die komplette Kontrolle über das System übernommen.

Ich hoffe, Sie haben erkannt, dass die Race-Condition weniger ein technischer Programmierfehler als viel mehr ein Fehler im logischen Design einer Applikation ist. Häufig sind es Seiteneffekte, die eine Race-Condition ermöglichen. Eine Verteidigung mit einer Firewall ist nicht möglich. Hier helfen nur Mandatory-Access-Control-Systeme wie SELinux, LIDS oder grsecurity. Wenn diese Systeme auch die

Race-Condition selbst nicht verhindern können, wenden sie jedoch weiteren Schaden von dem System ab.

### 4.4.7 SQL-Injektion

Die SQL-Injektion wird hier als ein Paradebeispiel eines Angriffs über einen Webserver beschrieben. Erwartungsgemäß erfolgen heute die meisten Angriffe über das Netzwerk entweder über das HTTP/HTTPS- oder das SMTP-Protokoll. Dies sind in vielen Fällen noch die einzigen Wege in das Netzwerk eines Unternehmens. Jeder andere Zugriff wird heute meist von Firewalls unterbunden. Mögliche Fernzugriffe werden über sichere VPN-Lösungen implementiert. Der Angreifer kann, vorausgesetzt Sie haben Ihre Hausaufgaben als Firewall-Administrator gemacht, nur auf den Webserver und den Mailserver des Unternehmens zugreifen. Allerdings bestehen die meisten Websites heute aus dynamischen Seiten, die von einer Web-Applikation erzeugt werden. Diese Web-Applikation greift hierzu auf eine Datenbank im Hintergrund zu. Über diesen Zugriff erhält die Web-Applikation eines Internet-Shops zum Beispiel Informationen über den Preis und die Verfügbarkeit eines Artikels. Diese Datenbank befindet sich meist in dem internen Netz des Unternehmens (siehe Abbildung 4.9). Der Webserver, der sich in der DMZ befindet, benötigt daher Zugriff auf diese Datenbank. Ein Angreifer, der diese Logik ausnutzen kann, erhält so unter Umständen Zugriff auf interne Systeme!

Bei der SQL-Injektion versucht der Angreifer herauszufinden, wie eine Web-Applikation die von ihm eingegebenen Daten verwendet. Wenn die Applikation

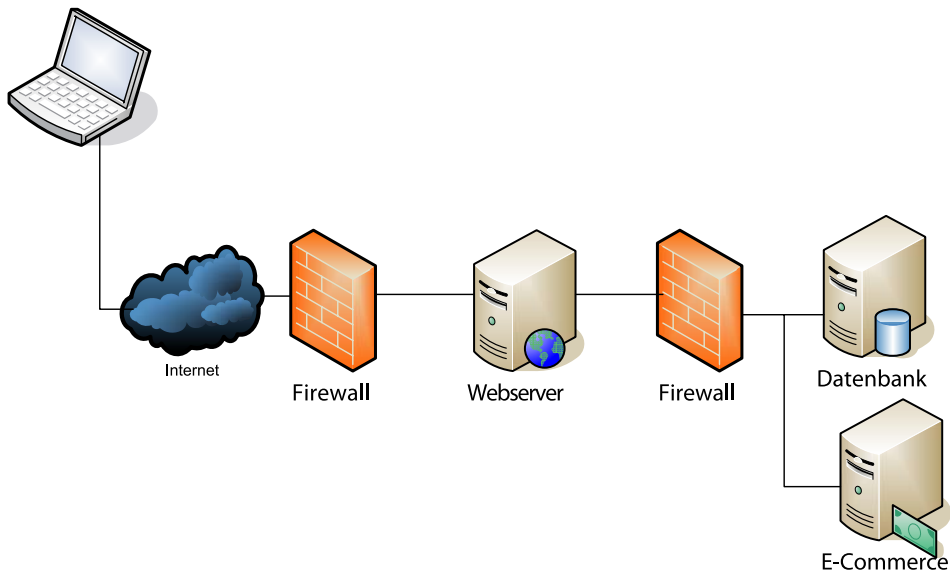


Abbildung 4.9: Die Web-Applikation auf einem Webserver greift meist durch die Firewall auf die interne Datenbank zu.



Abbildung 4.10: Viele Applikationen sind durch eine Login-Seite geschützt.

zum Beispiel über eine Login-Seite verfügt (Abbildung 4.10), dann besteht die Möglichkeit, dass die Applikation zur Überprüfung der Daten folgende Anfrage an die Datenbank stellt:

```
SELECT name FROM users WHERE name="$name" AND password="$pass"
```

Wenn der Angreifer als Namen `Bob` und als Kennwort `password` eingibt, wird der folgende SQL-Befehl ausgeführt:

```
SELECT name FROM users WHERE name="Bob" AND password="password"
```

Existiert der Benutzer und ist das Kennwort korrekt, so liefert diese Anweisung den Namen des angemeldeten Benutzers. Vielleicht kennt der Angreifer den Namen, aber nicht das Kennwort eines Benutzers. Dann kann er folgende Eingabe versuchen:

```
name=Bob
pass="weissnicht" OR "1"="1"
```

Wird dies nun in der SQL-Anweisung eingesetzt, ergibt sich die folgende Anweisung:

```
SELECT name FROM users WHERE name="Bob" AND
password="weissnicht" OR "1"="1"
```

Da der Ausdruck `1=1` immer stimmt, ist die Anmeldung unabhängig von dem eingegebenen Kennwort gültig, da die Select-Anweisung ein Ergebnis liefert.

Diese Sicherheitslücke ist möglich, da der Programmierer die eingegebenen Daten nicht vor der Verwendung prüft. Es handelt sich um eine fehlende Eingabevalidierung ähnlich dem Bufferoverflow oder dem Formatstring-Angriff. Die Web-Applikation muss sämtliche Eingaben auf ihre Gültigkeit prüfen, bevor diese an die

Datenbank weitergereicht werden. In diesem Beispiel wäre es zum Beispiel sinnvoll, beim Benutzernamen zu prüfen, ob außer den Zeichen A-Z,a-z weitere nicht-erlaubte Zeichen in dem Namen oder außer A-Z,a-z,0-9 weitere Zeichen in dem Kennwort vorkommen. Diese Aufgabe kann jedoch nur von der Web-Applikation selbst erledigt werden. Nur der Programmierer der Web-Applikation kann die gültigen Zeichen einer Eingabe festlegen. Eine Firewall ist hier überfordert.

### 4.4.8 Welchen Schutz bietet eine Firewall?

Ich habe Ihnen auf den letzten Seiten verschiedene Bedrohungsszenarien bei der Anbindung Ihrer Rechner an das Internet vorgestellt. Bei fast allen Angriffen habe ich Sie darauf hingewiesen, dass sie von einer Firewall nur in geringem Maße oder gar nicht abgewehrt werden können.

Warum sollen Sie dann überhaupt den Aufwand einer Firewall betreiben?

Ohne Firewall wird alles noch schlimmer! Überlegen Sie sich, welche Dienste Sie in Ihrem Netz und auf Ihren Rechnern betreiben. Sicherlich möchten Sie nicht, dass jeder im Internet auf diese Dienste zugreifen kann. Wahrscheinlich ist es auch nicht in Ihrem Sinne, dass jeder Benutzer auf jeden Dienst im Internet zugreifen kann. Hier bietet die Firewall Schutz. Natürlich erkennt die Firewall keinen Bufferoverflow- oder Formatstring-Angriff. Ohne Firewall kann ein Angreifer mit diesen Methoden alle Ihre Systeme angreifen. Wenn Sie eine Firewall besitzen, ist dieser Angriff nur auf den Systemen möglich, auf die die Firewall den Zugriff erlaubt.

Eine Firewall bietet Schutz nach dem Prinzip »Single-Point-of-Defense«. Anstatt jeden Rechner einzeln zu verteidigen und die Dienste in allen Punkten auf Sicherheitslücken zu prüfen, regelt die Firewall, wer auf welche Dienste zugreifen darf. Natürlich sind diese Dienste dann immer noch besonders gefährdet. Diese Dienste können immer noch mit einem Bufferoverflow-Angriff getroffen werden. Die tatsächliche Zahl der verwundbaren Systeme und Dienste haben Sie aber durch den Einsatz einer Firewall reduziert.

Eine Firewall hilft auch bei der Schadensbegrenzung nach einem erfolgreichen Angriff, denn Angriffe erfolgen nur selten blind. Der Angreifer will nach dem Bufferoverflow auf der Kommandozeile des Zielsystems arbeiten, seine Privilegien erweitern und weiteren Schaden anrichten. Hierfür benötigt er Netzwerkverbindungen, Werkzeuge und die Möglichkeit, weitere Systeme zu kontaktieren.

Betrachten Sie noch einmal die Abbildung 4.9. Wenn die Firewall Verbindungen des Webservers in das Internet unterbindet, kann ein Angreifer keine weiteren Werkzeuge nachladen. Auch der Start einer Remote-Shell, die eine neue Netzwerkverbindung aufbaut, ist nicht ohne weiteres möglich. Erlaubt die Firewall zusätzlich nur den Zugriff auf den Port des Datenbankservers in dem internen Netz, sind die weiteren Rechner vor dem Zugriff des Angreifers geschützt. Der Angreifer muss jetzt erst eine Sicherheitslücke in dem Datenbankserver finden, um diesen angreifen zu können.

Daher ist eine Firewall sehr wohl sinnvoll. Die Firewall schützt aber nicht vor allen Gefahren und Bedrohungen aus dem Internet. Sie müssen diese Bedrohungen kennen, um zusätzliche Maßnahmen zu implementieren, die Systeme regelmäßig zu patchen und so Sicherheit aufrechtzuerhalten.