Stefan Wohlfeil

Sicherheit im Internet 2

Kurseinheit 1:
Benutzersicherheit

mathematik
und
informatik

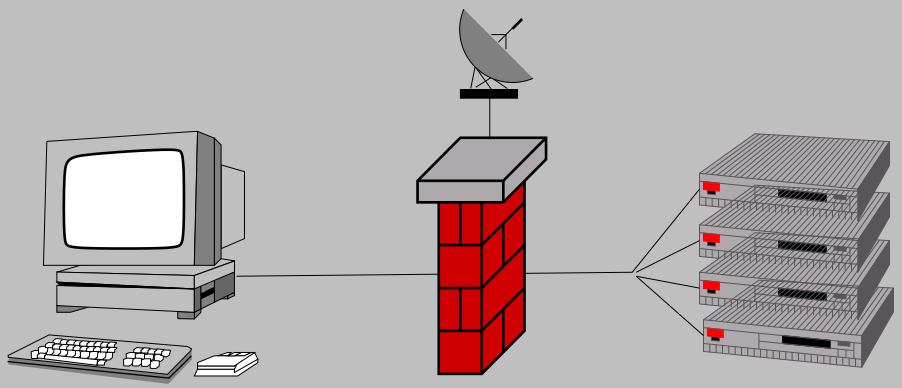
FernUniversität in Hagen

001 993 453 (10/08) 01867 - 4 - 01 - S 1

Kurs 01867: Sicherheit im Internet 2

Kurseinheit 1: Angriffe auf Rechner oder Netze

Autor: Stefan Wohlfeil



Kurs–Version 3

Inhalt

1 Angriffe auf Rechner oder Netze **1**
1.1 Einführung . . . 1
1.2 Sniffing . . . 3
1.3 Spoofing . . . 5
1.3.1 IP spoofing . . . 6
1.3.2 DNS spoofing . . . 6
1.4 Wörterbuchangriffe . . . 10
1.5 Buffer Overflow Angriffe . . . 11
1.6 URL hacking und Phishing . . . 15
1.7 Gefährliche Benutzereingaben . . . 17
1.7.1 HTML Code . . . 17
1.7.2 SQL injection . . . 18
1.8 Spezielle „Hacker“ Tools . . . 21
1.8.1 Host Scanner . . . 22
1.8.2 Network Scanner . . . 25
1.8.3 Hintertüren . . . 29
1.8.4 Zusammenfassung spezielle Tools . . . 32
1.9 Angriffe auf Verschlüsselung . . . 33
1.9.1 Klassifikation der Angriffe . . . 33
1.9.2 Brute Force Angriffe . . . 35
1.9.3 Lineare Kryptoanalyse . . . 37
1.9.4 Differentielle Kryptoanalyse . . . 38
1.9.5 Faktorisierung großer Zahlen . . . 38
1.9.6 Quantencomputer . . . 40
1.10 Zusammenfassung . . . 42
Literatur . . . 45

2 Benutzersicherheit **49**

3 Anbietersicherheit **77**

4 Rahmenbedingungen und Software-Prozesse **127**

Kurseinheit 1

Angriffe auf Rechner oder Netze

Die Autoren: Prof. Dr. Stefan Wohlfeil, geb. 12.12.1964

- Studium der Informatik mit Nebenfach Elektrotechnik an der Universität Kaiserslautern (1984–1991)
- Wissenschaftlicher Mitarbeiter am Lehrgebiet Praktische Informatik VI der FernUniversität Hagen (1991–1998)
- Promotion zum Dr. rer. nat. (1997)
- Mitarbeiter in der Deutsche Bank AG, Abteilung TEC — The Advanced Technology Group (1998–2002)
- Professor in der Fakultät IV, Abteilung Informatik der Fachhochschule Hannover; Arbeitsgebiet: Sichere Informationssysteme (seit 2002)

Einige Abschnitte in Kurseinheit 2 wurden von Prof. Dr. Jörg Keller (Lehrgebiet Parallelität und VLSI an der FernUniversität Hagen) geschrieben.

1.1 Einführung

Liebe Fernstudentin, lieber Fernstudent,
herzlich willkommen beim zweiten Kurs über Sicherheit im Internet!

Diese Einführung soll Ihnen einen Überblick darüber geben, worum es im vorliegenden Kurs geht. Dieser Kurs gehört zum Bereich *M5 Betriebssysteme, Verteilte und Kooperative Systeme* und ist ein Teil des Moduls *Sicherheit – Safety & Security.*[1] Der vorliegende Kurs vertieft das Thema *Sicherheit*, in das bereits im Kurs *(01866) Sicherheit im Internet I* eingeführt wurde.

Vorkenntnisse

Inhalt des Kurses und Vorkenntnisse: Dieser Kurs richtet sich an Informatik-Studierende und setzt die Kenntnis der Inhalte aus dem Informatik-Grundstudium voraus. Kenntnisse aus dem Kurs *(01866) Sicherheit im*

[1] Stand dieser Information August 2010. Beachten Sie bitte immer auch die Hinweise zur Belegung und die Informationen aus dem Prüfungsamt.

Internet I sind hilfreich. Konkret sollten Sie bereits wissen, wie ein Computer prinzipiell aufgebaut ist, was ein Betriebssystem typischerweise macht und welche Möglichkeiten sich durch die Vernetzung, wie beispielsweise im Internet, für Anwender bieten. Außerdem sollte Ihnen die Grundlagen des Internet und die grundsätzlichen Bedrohungen dort bereits bekannt sein.

Inhalt

Die Kurseinheit 1 beschäftigt sich mit den möglichen Angriffen gegen Computer oder Netze. Hier werden Techniken vorgestellt, mit denen Angreifer versuchen Schaden anzurichten. Dazu gehören Techniken wie Abhören von Datenpaketen, Vortäuschen falscher Tatsachen, Einsatz von „Hacker"-Programmen oder „Knacken" von verschlüsselten Nachrichten.

Bezahlverfahren

digitale Münzen

Hashing

Primzahlen

Zufallszahlen

Das Thema Benutzersicherheit wird in Kurseinheit 2 weiter vertieft. Dazu wird zunächst ein Überblick über wirtschaftliche Aspekte des Internets gegeben. Bei wirtschaftlichem Handeln geht es zumeist um Tausch, beispielsweise von Waren gegen Geld. Im Kurs werden verschiedene Bezahlverfahren vorgestellt, die im Internet eingesetzt werden können. Außerdem werden die technischen Grundlagen für die Realisierung von digitalem Geld vorgestellt.

Weiterhin werden in dieser Kurseinheit die Themen Hashing sowie Erzeugung von Primzahlen und Zufallszahlen vertieft. Gute und sichere Hashfunktionen sind im Bereich digitale Signaturen von enormer Bedeutung. Ohne solche Funktionen würden die im Kurs *(01866) Sicherheit im Internet I* vorgestellten Methoden nicht funktionieren. Sie lernen einige der mathematischen Hintergründe zu Hashfunktionen kennen sowie einige Aspekte zu den Themengebieten Primzahltests und Zufallszahlen.

VPN

IDS

In Kurseinheit 3 wird das Thema Anbietersicherheit vertieft. Die beiden vorgestellten Schwerpunkte sind (1) Virtual Private Networks (VPN) und (2) Intrusion Detection Systeme (IDS). Mit Hilfe von VPNs kann man Rechner an weit auseinanderliegenden Orten sicher über ein potentiell unsicheres Netz (z. B. das Internet) miteinander verbinden.

Mit Hilfe von Intrusion Detection Systemen sollen Administratoren Angriffe auf die eigenen Rechner, bzw. das eigene Netz erkennen. Die Funktionsprinzipien und die Konfiguration solcher Systeme werden in dieser Kurseinheit vorgestellt.

Gesetze

Am Ende der Kurses werden in Kurseinheit 4 gesetzliche und organisatorische Aspekte vorgestellt. Sie beginnt mit einem Überblick über relevante Gesetze und Verordnungen. Ihre Inhalte und die praktischen Auswirkungen werden, ohne in juristische Details zu gehen, vorgestellt.

Erstellung sicherer Systeme

Für einen Informatiker sind neben der Technologiekenntnis auch die Vorgehensmodelle bei der Erstellung sicherer Systeme von Bedeutung. Es werden Prozesse vorgestellt, die beim Bau sicherer Systeme hilfreich und nützlich sind. Den Abschluß des Kurses bilden Hinweise auf Systemarchitekturen.

Ergänzende Materialien: Das Thema Sicherheit im Internet ist derart umfangreich, daß es auch in diesem vertiefenden Kurs nur in Ausschnitten behandelt werden kann. Ziel des Kurses ist es, daß Sie die Grundlagen des Themen-

gebietes kennenlernen und Sie sich dann darauf aufbauend tiefer in die Materie einarbeiten können. Dazu gibt es verschiedene weitere Informationsquellen.

Bücher

Die Menge an Büchern zum Thema Security wächst sehr schnell. Ausgehend vom Literaturverzeichnis dieses Kurses sollten Sie in der Universitätsbibliothek das eine oder andere Buch ausleihen und durchschauen. Aktuellste Bücher kann man bei den verschiedenen Buchhändlern im Internet suchen. Dort findet man u. U. auch Rezensionen der Bücher vor.

Internet

Überhaupt ist das Internet eine nahezu unerschöpfliche Quelle an Informationen zum Thema Security. Im Kurs werden eine Reihe von Verweisen auf interessante Seiten im Internet genannt. Wenn Sie Zugang zum Internet haben, nehmen Sie sich doch die Zeit und schauen sich die eine oder andere Seite an. Ich hoffe, daß die Verweise noch stimmen, wenn Sie den Kurs lesen. Das Internet ändert sich ständig, so daß es gut sein kann, daß Sie einmal eine Seite nicht finden. In diesem Fall sollten Sie eine der vielen Suchmaschinen wie *altavista*, *bing*, *google*, *yahoo*, usw. konsultieren. Vielleicht hat sich ja nur die Adresse der Seite leicht verändert. Informieren Sie dann auch bitte die Kursbetreuer, damit der Kurstext aktualisiert werden kann. Die Namen und Sprechstunden der Kursbetreuer wurden Ihnen im Anschreiben zusammen mit dieser Kurseinheit genannt.

1.2 Sniffing

sniffer

Systeme, die Datenverkehr auf einem Netz mitlesen, nennt man **sniffer** von schnüffeln (engl. **to sniff**). Die Möglichkeiten um Datenverkehr zu belauschen hängen stark von der verwendeten Übertragungstechnologie ab. In Funknetzen (WLAN) ist es einfacher, die Signale zu empfangen als in drahtgebundenen Netzen. Im zweiten Fall braucht man zumindest einen physischen Zugang zu einer Leitung des Netzes.

hub

Lokale Netze, die mit einem **hub** realisiert sind, lassen sich vergleichsweise einfach belauschen. Ein hub verteilt alle ankommenden Datenpakete an alle angeschlossenen Geräte. Die Ethernet-Karten in PCs sind normalerweise so konfiguriert, daß sie Pakete, die nicht für diesen PC gedacht sind, verwerfen. Man kann eine Ethernet-Karte jedoch auch so konfigurieren, daß sie alle Pakete entgegen nimmt. Jedes Paket wird dann an die nächsthöhere Schicht im Protokoll-Stack geleitet. Dieser Betriebsmodus wird **promiscuous mode** genannt. Normalerweise benötigt man Administratorrechte, um diesen Modus einzuschalten.

promiscuous mode

Die so gelesenen Datenpakete werden von einem sniffer-Programm gesammelt und analysiert. Da die Analyse mit einigem Aufwand verbunden ist, werden die Pakete häufig erst einmal gesammelt, d. h. in eine lokale Datei geschrieben, und später analysiert. Dabei versucht der sniffer bekannte Protokolle, wie beispielsweise telnet, ftp oder http, in den gespeicherten Paketen zu entdecken. Ein Beispiel für die Ausgabe des Programms *ethereal*[2] zeigt Abbildung 1.1.

[2]Seit Juni 2006 existiert ein Folgeprojekt namens *wireshark*. Ethereal wird nicht mehr weiter entwickelt und durch das Nachfolgeprogramm *wireshark* ersetzt.

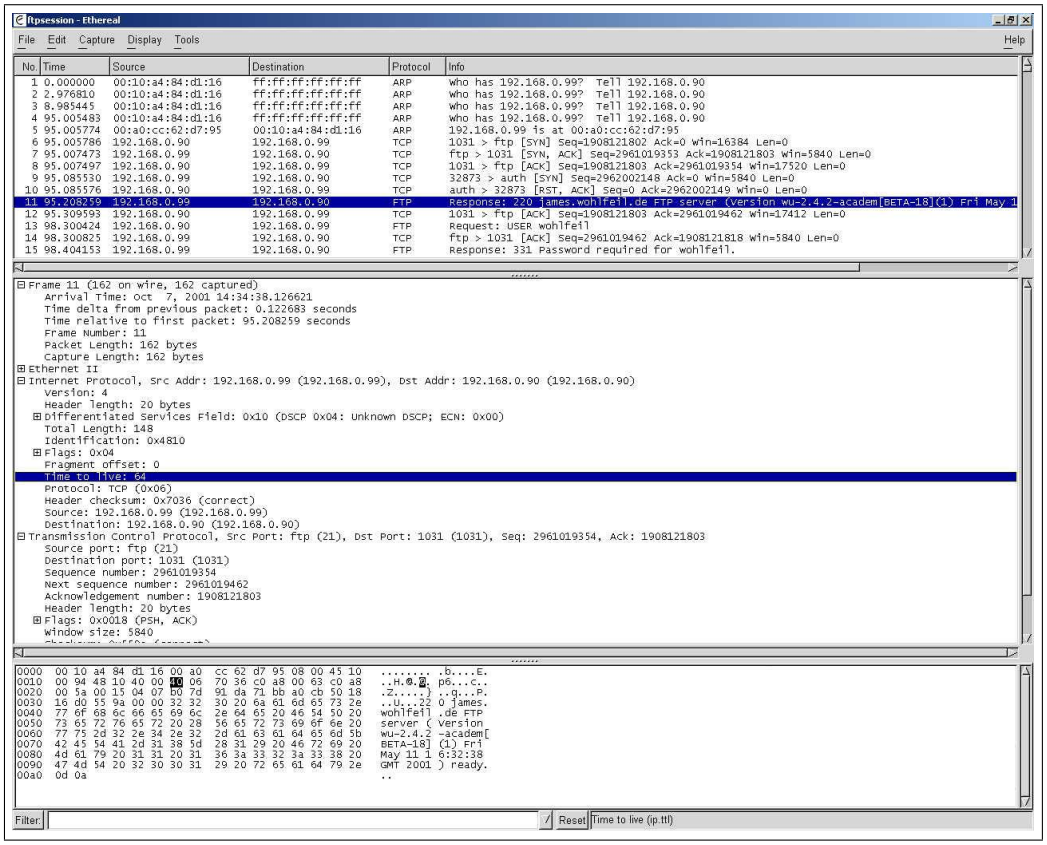


Abbildung 1.1: Ausgabe des Programms *ethereal*

Ganz unten in Abbildung 1.1 sieht man die Bytes des Datenpakets, einmal in hexadezimaler Darstellung und einmal als ASCII-Zeichen. Der Anfang des angezeigten Pakets enthält die Header-Informationen. Sie können normalerweise nicht sinnvoll als ASCII-Zeichen dargestellt werden. Statt dessen zeigt das Programm einen Punkt an. Weiter hinten im Datenpaket stehen dann die Informationen aus dem Rumpf. Sie sind häufig als ASCII-Zeichen kodiert und können somit auch vom Menschen gelesen und interpretiert werden. In Abbildung 1.1 enthält der Rumpf des Pakets die Begrüßungsmeldung eines ftp-Servers.

Das Ergebnis der genauen Analyse ist in der Mitte von Abbildung 1.1 dargestellt. Man sieht beispielsweise

- Quelladresse (Source),
- Zieladresse (Destination),
- Portnummern (Source Port, Destination Port) und
- weitere Verwaltungsdaten (Sequence number, Acknowledgement number, etc.)

Selektiert man eine Zeile im mittleren Bereich mit der Maus, so werden die zugehörigen Bytes im unteren Bereich invers dargestellt. Die hexadezimale Zahl 40 entspricht der Dezimalzahl 64 und der Wert an der invertierten Stelle ist das Lebensdauer-Byte, (engl. **time to live**).

Der obere Bereich von Abbildung 1.1 zeigt in einer Zeile zusammengefaßt die wichtigsten Informationen der einzelnen Pakete. Man erkennt, daß der Rechner mit der IP-Adresse 192.168.0.90 eine Verbindung zum Rechner mit der IP-Adresse 192.168.0.99 aufbauen will. Da die **MAC**-Adresse (engl. **Media Access Control**) beim Absender nicht bekannt ist, werden am Anfang **ARP**-Pakete (engl. **Address Resolution Protocol**) an alle angeschlossenen Systeme geschickt. Die hierfür reservierte Broadcast-Adresse besteht digital aus lauter Einsen, hexadezimal also ff:ff:ff:ff:ff:ff (siehe Spalte *Destination*). Anschließend findet der typische Handshake in drei Phasen (engl. **three way handshake**) zum Verbindungsaufbau statt. Er besteht aus (siehe Zeilen 6–8):

MAC

ARP

1. Einem syn-Paket vom Absender an den Zielrechner
2. Einem syn/ack-Paket vom Zielrechner an den Absender
3. Einem ack-Paket vom Absender an den Zielrechner

Danach wird die Begrüßungsmeldung vom Zielrechner, in diesem Fall einem ftp-Server, übertragen. In Zeile 13 sieht man, daß die Anmeldung am ftp-Server unter der Benutzer-Kennung `wohlfeil` erfolgt. Zeile 16 (absichtlich nicht in Abbildung 1.1 mit aufgenommen :-) enthält das Paßwort des Benutzers im Klartext.

Das Paßwort und die Benutzer-Kennung sind für einen Angreifer wahrscheinlich am wertvollsten. Mit ihnen kann er sich später auch am ftp-Server anmelden und vorgeben, der Benutzer `wohlfeil` zu sein. Allerdings findet man auch in den mitgelesenen Datenpaketen bereits viele weitere wichtige Informationen. Neben den Verzeichnisnamen und Dateinamen werden auch die angeforderten Dateien im Klartext übertragen. Sie können vertrauliche Informationen enthalten.

Dieses Programm funktioniert natürlich nur, wenn der Zugriff auf ein gemeinsames Übertragungsmedium gegeben ist. Ein solches Medium kann ein klassischer hub oder ein drahtloses lokales Netz (engl. **wireless LAN**) sein. Nur dann kann der Angreifer alle Datenpakete, also auch die, die nicht an seine Adresse gerichtet sind, mitlesen. Durch den Einsatz eines **Switches** kann man diese Art von Angriffen leider nicht völlig vereiteln, aber doch wenigstens erschweren.

Switches

Ein Switch verteilt eingehende Pakete normalerweise nicht an alle angeschlossenen Geräte, sondern nur an den im Paketkopf genannten Empfänger. Dazu muß der Switch natürlich wissen, welche Geräte mit welchen Adressen an seinen Anschlüssen eingesteckt sind. Dies lernt der Switch während des Betriebs, indem die Absenderadressen von eingehenden Paketen im Switch gespeichert werden. An dieser Stelle können nun weitere Angriffe einsetzen, siehe dazu Abschnitt 1.3.1.

1.3 Spoofing

Unter dem Begriff **Maskierungsangriff** (engl. **spoofing**) versteht man das

Maskierungsangriff

Vortäuschen einer falschen Identität. Dabei kann der Angreifer eine falsche IP-Adresse seines Systems oder einen falschen DNS-Namen seines Systems vortäuschen. In den folgenden Abschnitten werden beide Verfahren kurz vorgestellt.

1.3.1 IP spoofing

DHCP

Jeder Administrator eines Rechners kann eine falsche IP-Adresse angeben. Die IP-Adresse kann automatisch, z. B. von einem **DHCP**-Server (engl. **Dynamic Host Configuration Protocol**) vergeben werden oder sie wird vom Administrator im System konfiguriert. Somit kann ein Administrator jederzeit problemlos eine IP-Adresse aus dem aktuellen Subnetz vergeben.

Router

Wie in Abbildung 1.1 gesehen, gehen ARP-Pakete an alle angeschlossenen Geräte. Das hat zur Folge, daß alle Geräte nach dem ersten Broadcast die MAC-Adresse des Absenders kennen. Ein Angreifer kann nun gefälschte ARP-Pakete verschicken. Das kann eine gefälschte Antwort auf eine ARP-Anfrage oder einfach eine gefälschte ARP-Nachricht sein. Besonders subtil wird dieser Angriff, wenn der Angreifer vorgibt, der **Router** zu sein. An den Router werden alle Pakete geschickt, deren Ziel nicht im lokalen Netz liegt. Der Angreifer muß allerdings darauf achten, die abgefangenen Pakete anschließend an den richtigen Empfänger weiter zu leiten. Andernfalls würden Pakete verloren gehen und der Absender könnte den Angriff bemerken.

1.3.2 DNS spoofing

Rechner im Netz werden normalerweise nicht über ihre MAC-Adresse direkt angesprochen, sondern über IP-Adressen. Numerische IP-Adressen können sich Menschen aber schlechter merken als Namen (Zu welchem Rechner gehört beispielsweise die Adresse 195.71.11.67?[3]). In der Regel werden daher symbolische Namen benutzt. Für das Internet ist ein hierarchischer Namensraum definiert.

domain

top level domains

Man kann sich diesen Namensraum als Baum vorstellen, an dessen Wurzel eine namenlose Wurzel-Domäne steht. Der Begriff **domain** bezeichnet immer einen Teilbaum im hierarchischen Namensraum. Unter der Wurzel gibt es dann die sogenannten **top level domains** wie beispielsweise `de`, oder `com`, usw. Unter diesen Domains gibt es dann Unterdomänen, usw. Einen kompletten DNS-Namen liest man von unten nach oben und trennt die Domain-Namen durch einen Punkt.

Domain Name Service

forward zone

reverse zone

Die Zuordnung von Namen zu IP-Adressen wird vom **Domain Name Service** (DNS) vorgenommen. Ein DNS-Server verwaltet zwei Datenbanken/Tabellen. Die Abbildung von Name auf IP-Adresse wird in der **forward zone** Tabelle gespeichert. Die umgekehrte Abbildung von IP-Adresse auf Name wird in der **reverse zone** Tabelle verwaltet. Das Design von DNS verlangt keine Maßnahmen zur Überprüfung der Konsistenz dieser Tabellen. Weiterhin findet in der Regel keine sichere Authentisierung bei der Erstellung oder dem Austausch der Informationen in den Tabellen statt.

[3] Am 16.8.2010 gehörte diese Adresse zum Namen `www.spiegel.de`

Probleme

Gefälschte oder inkonsistente Einträge können zu folgenden Problemen führen:

1. Unter UNIX kann ein Benutzer/Administrator anderen Rechnern trauen und einen Verbindungsaufbau von diesen Rechnern *ohne* explizite Authentisierung erlauben. Die Kommandos `rlogin` oder `rsh` haben diesen Mechanismus implementiert. Die Spezifikation der vertrauenswürdigen Rechner geschieht über DNS-Namen.

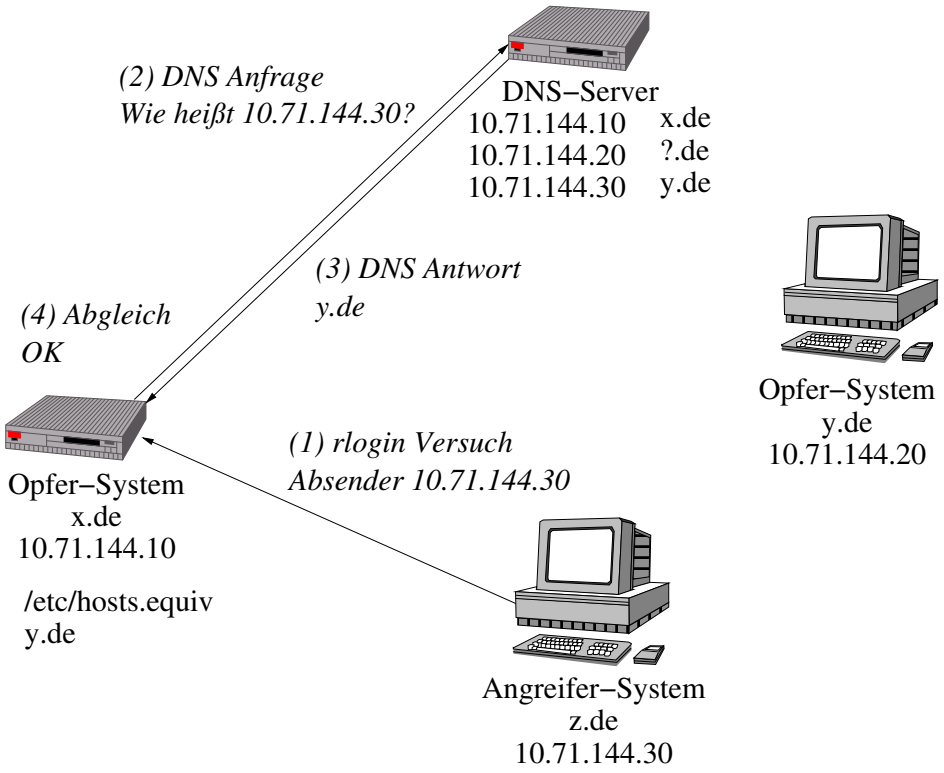


Abbildung 1.2: Angriff auf reverse zone

Das folgende Beispiel (siehe Abbildung 1.2) zeigt, welche Bedingungen erfüllt sein müssen und was beim Verbindungsaufbau genau passiert. Auf dem Rechner `x.de` soll dem Rechner `y.de` vertraut werden. Dazu wird auf dem Rechner `x` in der Datei `/etc/hosts.equiv` der Name `y.de` eingetragen.

Der Angreifer `z.de` soll nun versuchen, eine Verbindung mit `x.de` aufzunehmen und dabei vorgeben, er sei Rechner `y.de`. Dazu muß der Angreifer die reverse zone Tabelle des DNS-Servers verändern. Wenn `z.de` das `rlogin` Kommando ausführt, dann steht in den IP-Paketen an `x.de` die echte IP-Adresse von `z.de` als Absender. `x.de` fragt den DNS-Server nach dem Namen zur Absender IP-Adresse. Liefert der DNS-Server nun (fälschlicherweise) den Namen `y.de` zurück, so prüft `x.de` nur noch, ob dieser Name in `/etc/hosts.equiv` steht. Falls ja, so wird die Verbindung ohne Authentisierung aufgebaut.

Bei diesem Angriff gibt sich der Angreifer als falscher Absender aus.

2. Schafft es ein Angreifer, die forward Tabelle des DNS-Servers mit falschen Einträgen zu füllen, so kann er fälschlicherweise das Ziel von Verbindungsaufbauwünschen werden. So könnte der Angreifer eine komplette Website auf dem eigenen Rechner nachbauen, beispielsweise einer E-commerce Firma, einer Behörde oder einer Bank.

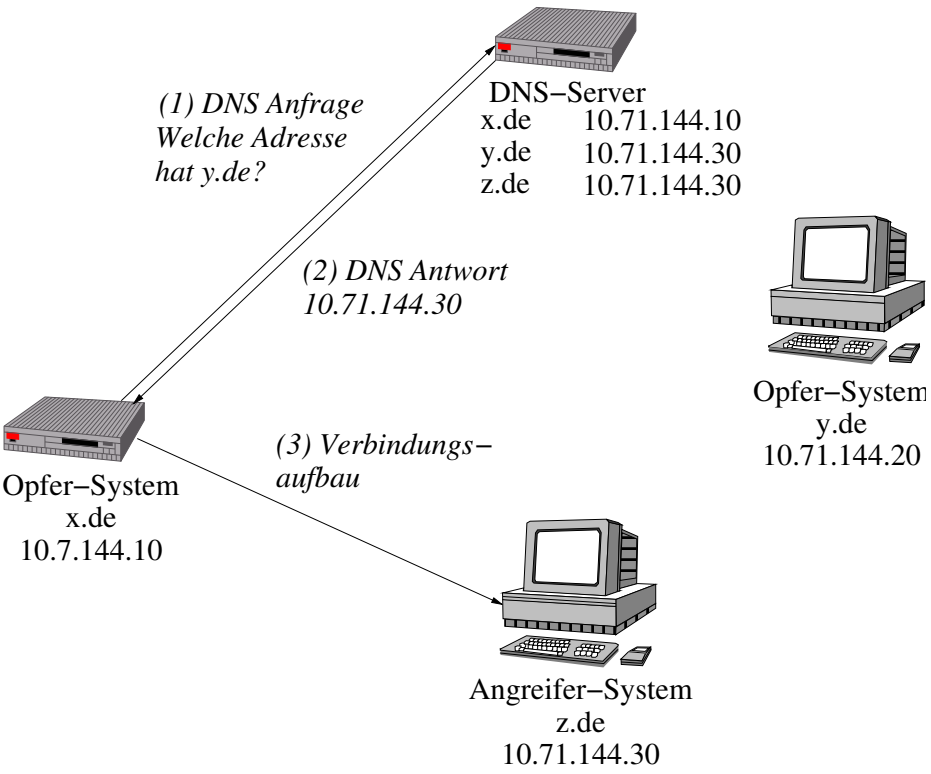


Abbildung 1.3: Angriff auf forward zone

Abbildung 1.3 zeigt den Ablauf dieses Angriffs. Der Angreifer `z.de` hat die forward zone Tabelle des DNS-Servers so manipuliert, daß seine IP-Adresse unter dem Namen des Opfers (`y.de`) eingetragen ist. Der Rechner `x.de` möchte nun eine Verbindung zu `y.de` aufbauen und erfragt beim DNS-Server die IP-Adresse von `y.de`. Der DNS-Server liefert die Adresse von `z.de` und `x.de` baut eine falsche Verbindung auf.

Dieser Angriff kann drei Schaden-Typen zur Folge haben: (1) *Reputationsschaden:* Der Angegriffene könnte durch willkürliche Änderungen an den Inhalten verunglimpft werden. (2) *Umsatzschaden:* Durch Umleiten von Datenverkehr kann im E-commerce Umsatz (und damit auch Ertrag) verhindert werden. Leitet ein Buchhändler beispielsweise den Verkehr von `amazon.de` auf seine Rechner, so erschleicht er sich dadurch möglicherweise neue Geschäfte. (3) *Vertraulichkeitsschaden:* Ist die Website interaktiv, d.h. es erfolgen Benutzereingaben, so kann der Angreifer an vertrauliche Informationen gelangen. Beim Internet-Banking gibt der Kunde seine **PIN** oder **TAN** ein und geht davon aus, daß diese Informationen nur an die Bank übertragen werden und nicht an einen Dritten.

PIN
TAN

Für die korrekte „Funktion des Internets“ ist ein korrekt arbeitendes DNS eine

unverzichtbare Voraussetzung. Internet Service Provider (ISPs) oder Staaten können durch Manipulationen des DNS die Internetnutzung für ihre Kunden, bzw. Einwohner verhindern. Möchte man den Zugriff auf unliebsame Webseiten verhindern, so ändert man einfach im DNS die IP-Adresse zum unliebsamen Namen auf einen anderen Wert. Surfer, die `http://www.unliebsam.xy` eingeben werden dann nicht mit dem unliebsamen Rechner verbunden, sondern bekommen eine andere Seite angezeigt. Natürlich können Benutzer immer noch durch die Angabe der IP-Adresse des Rechners in der URL die Seite abrufen. Dazu muß man die IP-Adresse erst einmal kennen. Außerdem bekommt man ein Problem, wenn man weitere Seiten durch Anklicken von URLs aufrufen will. Die links in HTML-Seiten enthalten i. d. R. die DNS-Namen der Rechner und nicht die IP-Adressen.

Da DNS-Anfragen und DNS-Antworten weder verschlüsselt, noch signiert sind ist ihre Authentizität und ihre Intgrität nicht gesichert. Ein Angreifer muß keinen DNS-Server manipulieren, sondern er kann auch die Nachrichten manipulieren. Um diese Art von Angriff zu verhindern, wurde eine Erweiterung des DNS-Protokolls definiert. Sie heisst **DNSSEC**. Die zugrundeliegende Idee besteht darin, daß DNS-Daten (konkret die Informationen zu einer DNS zone) digital signiert werden und jeder Empfänger durch Überprüfen der Unterschriften prüfen kann, ob die Daten authentisch und integer sind.

DNSSEC

Die Herausforderung bei der Einführung von DNSSEC liegt darin, daß DNS ein verteiltes System ist. Alle DNS-Server müssten es benutzen, d. h. die Server-Software muß aktualisiert werden und Schlüsselpaare für die Signierung müssen erstellt werden. Wie in public key Infrastrukturen üblich muß man dann auch noch sicherstellen, daß man den korrekten öffentlichen Schlüssel der Gegenseite besitzt. Nur dann kann man die digitalen Signaturen prüfen.

In DNSSEC sind zwei Schlüsseltypen vorgesehen:

1. **key signing keys**. Diese Schlüssel werden nur dazu benutzt, andere Schlüssel digital zu signieren. Signiert beispielsweise die Wurzeldomäne (engl. **root domain**) mit ihrem privaten Schlüssel den Schlüssel einer der obersten DNS-Domäne (engl. **top level domain**), wie z. B. die `.de`-Domäne, dann bedeutet das, daß die Top-level-Domäne den Verantwortlichen für die .de-Domäne vertrauen und die .de-Domänendaten mit dem entsprechenden Schlüssel signiert sind.

   key signing keys

2. **zone signing keys**. Mit diesen Schlüsseln werden die eigentlichen DNS-Daten signiert. Eine zone bezeichnet hier einen Ausschnitt aus dem DNS-Namensraum, nämlich den Ausschnitt, für den ein DNS-Server zuständig ist. Dieser DNS-Server liefert also die „gültigen“ (engl. **authoritative**) Antworten zu Anfragen zu dieser Zone. Der Teilbaum `inform.fh-hannover.de` ist beispielsweise eine Zone und der DNS-Server der Abteilung Informatik ist für diese Zone zuständig. Der Administrator richtet neue Rechner ein, weist IP-Adressen und DNS-Namen zu und trägt diese Informationen in den DNS-Server ein.

   zone signing keys

Damit eine Antwort nun überprüft werden kann, braucht der Prüfende den öffentlichen Schlüssel des Unterzeichners. Einen solchen öffentlichen Schlüssel

gibt es für *jede* DNS-Zone. Damit man nun nicht alle diese öffentlichen Schlüssel kennen muß, kann man auch eine Vertauenskette (engl. **chain of trust**) benutzen. Kennt man den öffentlichen key-signing-Schlüssel einer Domäne (z. B. `fh-hannover.de`), so kann man den DNS-Daten einer Unter-Domäne (z. B. `inform.fh-hannover.de`) vertrauen, sofern sie mit einem Schlüssel signiert sind, der vom o. g. key-signing-Schlüssel signiert ist.

Im Juli 2010 wurden die Root-Server des DNS auf DNSSEC umgestellt. Als nächstes müssen die Betreiber der Top-Level-Domänen ihre Systeme auf DNSSEC umstellen, dann die der Unterdomänen, usw. Am Ende müssen die DNS-Benutzer dann „nur noch“ die key-signing-Schlüssel der Top-level-Domäne kennen und können dann die DNS-Daten überprüfen.

Weitere Informationen zu DNSSEC finden Sie bei Kolkman [Kol09] oder in den RFCs zu DNSSEC. Als Startpunkt dient dort RFC 4035 *Protocol Modifications for the DNS Security Extensions*. Aktuell (Stand August 2010) gibt es über 15 RFCs in deren Titel das Wort DNSSEC vorkommt.

1.4 Wörterbuchangriffe

Bei diesem Angriffstyp versucht der Angreifer, das Paßwort eines legitimen Benutzers durch systematisches Ausprobieren herauszufinden. Dabei gibt es zwei verschiedene Strategien:

Strategien

1. Ausprobieren von bekannten Wörtern und ihren Abwandlungen, bzw. Kombinationen.

2. Ausprobieren aller möglichen Zeichenketten.

Die erste Methode setzt darauf, daß Benutzer ein Paßwort wählen das sie sich einfach merken können. Oft sind das Wörter, die auch in Wörterbüchern wie dem *Duden* vorkommen. Im Internet existieren Wörterbücher in elektronischem Format für verschiedene Sprachen (Deutsch, Englisch, etc.) oder zu verschiedenen Themengebieten (Biologie, Technik, etc.). Programme können diese Wörter systematisch ausprobieren. Auch können dabei Variationen, wie veränderte Groß-/Kleinschreibung, oder Ersetzungen (z. B. Buchstabe o durch Ziffer 0) mitgeprüft werden.

brute force

Die zweite Methode zählt alle möglichen Zeichenketten auf und prüft sie. Dabei werden natürlich auch alle bekannten Wörter vorkommen, jedoch besteht die Mehrzahl überwiegend aus „unsinnigen“ Zeichenketten. Bei einer Größe des Alphabets von x und einer Wortlänge von n gibt es x^n verschiedene Wörter. Für $x = 64$ Zeichen und eine Länge von 8 sind dies bereits 64^8.

Aufwand

Dauert jeder Test nun 1 Millisekunde, so braucht man ca. $64^8/1000 \approx 281$ Milliarden Sekunden; das entspricht etwa 3.25 Millionen Tagen. Bei ausreichend langen Paßwörtern ist diese Angriffsart nicht mehr praktikabel.

Bei beiden Methoden wird eine Hash-Funktion auf das zu prüfende Wort angewendet (siehe z. B. Kurs *(01866) Sicherheit im Internet I* oder Kurs *(01801) Betriebssysteme und Rechnernetze*). Die Hash-Funktion ist dieselbe, die auch das Betriebssystem auf ein Paßwort anwendet, bevor es das

Paßwort speichert. Der Hash-Wert jedes zu prüfenden Wortes wird mit jedem gespeicherten Hash-Wert verglichen. Dazu ist natürlich der Zugriff auf die gespeicherten Hash-Werte erforderlich. Unter Windows NT und seinen Nachfolgern gibt es drei Möglichkeiten, um an die Hash-Werte der Paßwörter zu kommen:

1. Auszug aus der Windows Registry erstellen:

   Mit Hilfe spezieller Hilfsprogramme (z. B. *pwdump*) kann man die Hash-Werte aus der Systemdatenbank (Registry) lesen. Dafür sind allerdings Administrator-Rechte sowie evtl. direkter, physischer Zugang zu dem Rechner erforderlich.

2. Auszug aus der SAM-Datei erstellen:

   Windows speichert die Benutzerdaten einschließlich Hash-Wert des Paßworts in der SAM-Datei. Der Zugriff auf diese Datei ist bei laufendem Windows allerdings nicht möglich. Das Betriebssystem hat diese Datei geöffnet und exklusiv für sich reserviert. Somit kann man die Datei weder lesen noch kopieren.

   Allerdings wird diese Datei mit gesichert. Man kann also eine Sicherung erstellen und daraus dann die Datei kopieren. Die Sicherung wird über das Betriebssystem selbst erstellt, so daß die oben genannte exklusive Reservierung ohne Bedeutung ist.

3. Abhören des Netzverkehrs:

   Bei der Anmeldung an einem Windows Client PC gibt der Benutzer sein Paßwort ein. Die Prüfung des Paßworts wird vom **Domain Controller** durchgeführt. Der Domain Controller ist eine Server Maschine, auf der unter anderem auch die Benutzerdaten verwaltet werden. Die Kommunikation zwischen Windows Client und Domain Controller ist im **Server Message Block (SMB)** Protokoll geregelt. Dabei kann das Paßwort verschlüsselt oder unverschlüsselt übertragen werden. Das Programm *Cain & Abel* kann beispielsweise den Netzverkehr mitschneiden und gleichzeitig versuchen, die Paßwörter zu knacken.

   Domain Controller

   Server Message Block (SMB)

Unter UNIX stehen die Benutzer-Kennung und die Hash-Werte der Paßwörter in der Datei `/etc/passwd`. Diese Datei darf jeder lesen. Um einen Wörterbuchangriff auf die Paßwörter zu verhindern, werden die Hash-Werte der Paßwörter heute in einer getrennten Datei, der sogenannten Shadow-Datei gespeichert. Sie kann nicht mehr von jedem gelesen werden.

1.5 Buffer Overflow Angriffe

Ein Speicherüberlauf (engl. **buffer overflow**) ist ein häufiger Grund für Systemabstürze. Dabei ist in einem Programm ein Speicherbereich fester Größe definiert, in den dann Daten *ohne* Längenprüfung geschrieben werden. Sind diese Daten länger als der Speicherbereich, so werden andere Teile des Speichers überschrieben. Dadurch können verschiedene Fehler auftreten.

Fehler

Programmabsturz: Falls der überschriebene Bereich binären Programmcode enthielt, der nun durch andere Zeichen überschrieben wurde, so ist die Wahrscheinlichkeit groß, daß diese Zeichen kein gültiger Maschinencode für die CPU sind. In diesem Fall stürzt das Programm einfach ab, sobald der Code im überschriebenen Bereich ausgeführt wird.

Ausführen anderen Programmcodes: Falls der Angreifer eigenen gültigen Maschinencode an die Stelle des bisherigen Codes setzt, dann wird dieser neue Maschinencode ausgeführt. In diesem Fall tut das Programm etwas anderes als es normalerweise machen würde.

Modifikation von Daten: An der Stelle, an der der Speicherüberlauf auftritt, können auch Daten stehen, die verändert werden und dann zu inkonsistentem Verhalten des Programms führen oder falsche Ausgaben des Programms erzeugen.

Beispiel

Ein Beispiel für den dritten Fall (aus [Bie00]) ist:

```
/* Hier das Passwort aus der Datenbank lesen */
char    origPasswort[12] = "Geheim\0";
char    userPasswort[12];

/* liest Benutzereingabe vom Terminal */
gets(userPasswort);

if (strncmp(origPasswort, userPasswort, 12) != 0)
{
        printf("Falsches Passwort!\n");
        exit(-1); /* Programm beenden */
}
/* Benutzer authentisiert */
```

Durch die Eingabe von mehr als 12 Zeichen kann der Benutzer in diesem Beispiel die Variable `origPasswort` überschreiben. Das gilt unter der Annahme, daß die Adresse von userPasswort[12+i] mit der Adresse origPasswort[i] übereinstimmt. Sie ist bei der Stack-Belegung der meisten Computer erfüllt. Falls nicht, dann existiert die Schwachstelle wenn die Reihenfolge der Variablendefinitionen vertauscht wird. Gibt der Benutzer nun genau 24 Zeichen ein, bei denen die vordere und die hintere Hälfte identisch sind, also beispielsweise „oberprogrammoberprogramm“, dann wird der Vergleich in der IF-Anweisung positiv ausfallen. Der Angreifer würde authentisiert und das Programm verhält sich anders als es sollte.

Ursache

Der Grund hierfür ist, daß prozedurale Programmiersprachen einen Laufzeitstapel (engl. **runtime stack**) (siehe Abbildung 1.4) benutzen. Der Hauptspeicher wird linear adressiert, d. h. jede Speicherzelle bekommt eine Adresse zwischen 0 und *Speichergröße*. Der Platz für statische Daten wird an einem Ende des Adreßraumes freigehalten, so daß ein zusammenhängender Adreßbereich für die dynamischen Daten übrig bleibt. Häufig gibt es zwei „Typen“ von dynamischen Daten:

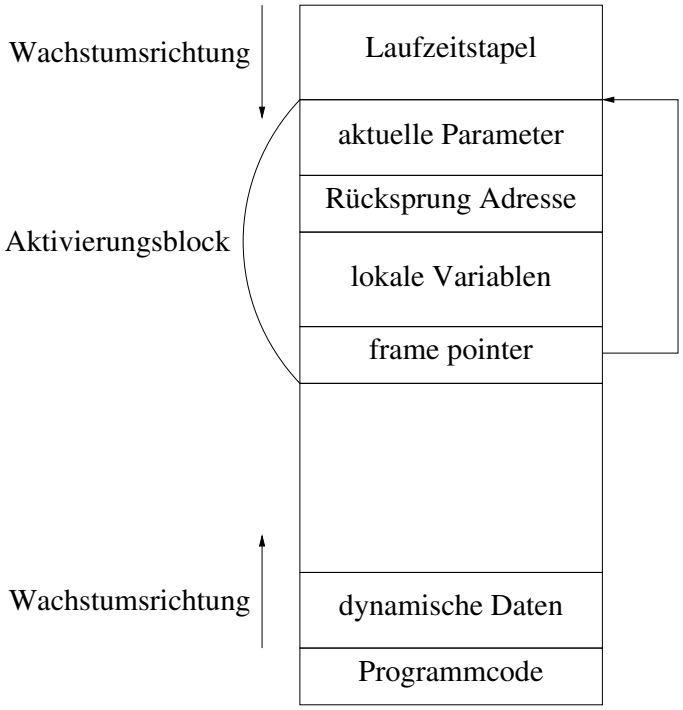


Abbildung 1.4: Hauptspeicheraufbau in prozeduralen Sprachen

1. Vom Programm zur Laufzeit *explizit* angeforderte Speicherbereiche: In prozeduralen Sprachen wie Modula 2 oder C existieren Systemfunktionen wie `ALLOCATE` oder `malloc`, mit denen Speicherbereiche angefordert werden können. In Objekt-orientierten Sprachen wie C++ oder Java wird neuer Speicher für Objekte durch den Operator `new` angefordert.

2. Vom Programm zur Laufzeit *implizit* benötigte Speicherbereiche: Hierzu gehört der Speicherplatz, der beispielsweise bei einem Funktionsaufruf benötigt wird.

Da der Platzbedarf für diese Bereiche nicht im voraus fest steht, organisiert man den Speicher wie folgt: Der eine Typ von Daten wächst vom unteren Ende des freien Speichers in Richtung Mitte, während der andere Bereich vom oberen Ende Richtung Mitte wächst. Hierbei hofft man natürlich, daß zur Laufzeit des Programms diese Bereiche nicht überlappen.

Zur Laufzeit eines Programms werden Funktionen aufgerufen, in einem C-Programm beginnt der Ablauf beispielsweise mit der Funktion `main()`. Für jede aufgerufene Funktion wird ein **Aktivierungsblock** (engl. **stack frame**) auf dem Laufzeitstapel angelegt. Darin ist Platz für die aktuellen Parameter der Funktion, die Rücksprungadresse, die lokalen Variablen und einen Zeiger auf den vorherigen Aktivierungsblock. Der Zeiger auf den vorherigen Aktivierungsblock ist erforderlich, um in Sprachen wie PASCAL oder Modula 2 auf Variablen in umschließenden Prozeduren zugreifen zu können. Die Rücksprungadresse enthält die Anweisung, mit der nach dem Ende der Funktion fortgefahren werden soll.

Aktivierungsblock

An dieser Stelle kann ein Angreifer nun versuchen, eigenen Programmcode zur Ausführung zu bringen. Dazu muß er zwei Dinge erreichen:

1. Den Programmcode, den der Angreifer ausführen möchte, muß er in den Adreßraum des Programms kopieren. Hierfür bietet sich der Speicher für lokale Variablen auf dem Laufzeitstapel an.

2. Die Rücksprungadresse muß auf den Speicherbereich zeigen, in den der Angreifer seinen Code kopiert hat.

Um das zu erreichen, muß der Angreifer den Prozessortyp des anzugreifenden Systems kennen und die entsprechenden Maschinencodes eingeben können. Unter Linux auf *Intel* Prozessoren kann man beispielsweise in unter 50 Bytes Länge Programmcode unterbringen, der eine Shell startet. Der Angreifer könnte also anschließend alle Kommandos auf dem Rechner ausführen.

Die folgenden System-/Bibliotheksfunktionen sind mögliche Ursachen für einen Speicherüberlauf. Sie sollten bei der Programmierung durch andere Funktionen ersetzt werden.

Unsichere Funktion	sicherere Alternative
gets	fgets
scanf	(Größenbegrenzung im Format-Tag)
sprintf	snprintf
strcpy	strncpy
strcat	strncat

Diese Liste ist bei weitem nicht vollständig. Viega und McGraw [VM01] haben ein ganzes Kapitel dem Thema *Buffer Overflow* gewidmet und darin eine Tabelle mit gefährlichen Funktionen, dem Grad der Gefährlichkeit und möglichen Lösungen des Sicherheitsproblems angegeben. Auch Hoglund und McGraw [HM04] haben sich ausführlich mit Buffer Overflows und wie man sie in verschiedensten Umgebungen ausnutzen kann beschäftigt.

Neben dem falschen Einsatz dieser Funktionen können Programmierer aber auch den Fehler machen, Daten in einer Schleife zu lesen und dabei die Längenprüfung „vergessen“. Ein typisches Code-Beispiel hierfür ist:

```
int     zeichen, i;
char    puffer[42];

i = 0;
while ( (zeichen = getc(stdin)) != '\n')
{
        puffer[i] = zeichen;
        ...
        i++;
}
```

Der Programmierer möchte alle Zeichen einer Zeile lesen und hat in seinem Speicherbereich aber nur Platz für 42 Zeichen. Ist die Eingabe länger, so tritt ein buffer overflow auf.

Gegenmaßnahmen

Buffer Overflow Fehler stecken in vielen Programmen. Die einzige Chance um diesen Fehlern zu entgehen besteht darin, die Programme entweder sorgfältiger zu schreiben oder bereits existierende Programme besonders auf diese

Fehler hin zu untersuchen. Bei der Untersuchung können spezielle Hilfsprogramme die Quelltexte automatisch nach „verdächtigen“ Stellen durchsuchen und dem Programmierer einiges an Arbeit abnehmen. Die genaue Überprüfung der gefundenen Stellen und die evtl. erforderliche Korrektur bleibt aber dem menschlichen Programmentwickler vorbehalten.

Man kann Buffer Overflow Fehler auch durch die Wahl einer anderen Programmiersprache verhindern. Sprachen wir Java oder C# wurden so definiert, daß Speicherbereiche genauer kontrolliert werden und somit Buffer Overflows nicht mehr möglich sind.

1.6 URL hacking und Phishing

In diesem Abschnitt geht es um einen Angriffstyp, bei dem wieder eine falsche Identität, bzw. eine falsche Absicht erzeugt wird. Dazu werden beispielsweise kleine Fehler (Flüchtigkeitsfehler) ausgenutzt. Zu dieser Kategorie Fehler gehören Tippfehler wie Buchstaben- oder Zahlendreher. Ein Angreifer nutzt dies aus, indem er DNS-Namen registriert, die leichte Abwandlungen bekannter Namen sind. Ein Beispiel hierfür ist der Name `wwwgmx.de`.

Abbildung 1.5: Kleine Änderung, große Wirkung

Er sieht auf den ersten Blick aus wie der Name eines bekannten Email Services. Allerdings fehlt der Punkt hinter dem Präfix `www`, so daß der DNS-Server hier zwei völlig verschiedene IP-Adressen liefert. Am 17. Juli 2002 waren dies:

Name	IP-Adresse	Zweck
www.gmx.de	213.165.65.100	Email Service
wwwgmx.de	212.227.119.94	Kontaktanzeigen

Ein Benutzer, der häufig seine URLs beim Surfen selbst eintippt, kann somit schnell auf einem anderen System landen. Nicht immer erkennt man den Fehler so schnell wie in oben genanntem Beispiel. Hätte der Angreifer auf seinem Rechner die Website des Opfers etwas genauer nachgebaut, so könnte er erheblichen Schaden verursachen. Falls das Opfer eine Bank oder Behörde wäre, so könnte der Angreifer vertrauliche Daten erfahren. Hierzu gehört eine mögliche Benutzer-Kennung und das zugehörige Paßwort.

Das gezeigte Beispiel gehört allerdings nicht in diese Kategorie. Statt dessen geht es dem Urheber in erster Linie darum, Benutzer im WWW auf seine Seite zu locken. Möglicherweise verdient er mit der dort dargestellten Werbung Geld und wird pro Besucher oder pro Seitenaufruf bezahlt.

Heute (August 2010) funktioniert das Beispiel aus Abbildung 1.5 nicht mehr. Betreiber großer Web-Sites sind dazu übergegangen, auch ähnliche Namen für sich zu registrieren. Sollte ein ähnlicher Name bereits registriert sein und möglicherweise mit kommerziellen Inhalten versehen sein, dann können die Inhaber des „Original"-Namens auf Unterlassung und Herausgabe des ähnlichen Namens klagen. Die Domäne `wwwgmx.de` gehört inzwischen auch zum bekannten Anbieter von Email-Diensten. Eine IP-Adresse gehört damit zu mehreren DNS-Namen, die man als Synonyme für den dahinter stehenden Web-Auftritt betrachten kann. Das Online-Portal des Bayerischen Rundfunkes erreicht man z. B. unter den Namen `www.bayerischer-rundfunk.de`, `www.br-online.de` oder `www.bronline.de`.

Suchmaschinen manipulieren

Um möglichst viele Besucher auf die eigenen Seiten zu locken, gibt es weitere Möglichkeiten. Hierzu gehört die Manipulation von Suchmaschinen, so daß man bei möglichst vielen Anfragen weit oben in der Trefferliste steht. Dazu müssen bestimmte Schlüsselwörter auf der Seite vorkommen. Die Suchmaschine indiziert diese Schlüsselwörter und bewertet die Seite entsprechend. Ruft ein Benutzer die Seite im Browser dann auf, so findet er keines der Schlüsselwörter mehr, sondern etwas völlig anderes. Die möglichen Gründe hierfür sind, daß das Schlüsselwort

- nur in einem Meta-Element vorkommt, dessen Inhalt niemals vom Browser angezeigt wird,
- in weißer Schrift auf weißem Grund steht,
- in einem nicht druckbaren Bereich steht.

Eine andere Möglichkeit ist es, daß das Schlüsselwort zwar auf der Seite vorkommt, aber in einem anderen Zusammenhang benutzt wird. Schneier [Sch00] nennt hierzu folgendes Beispiel: „Dieser billige Pullover (nicht Prada, nicht Armani) ist rot.[4]"

Phishing-Angriffe

Eine Variante dieser Angriffsform sind die sogenannten **Phishing-Angriffe**. Ein Angreifer baut die HTML-Seiten, z. B. einer Bank, realistisch und echt aussehend auf seinem Server nach. Für seinen Server registriert der Angreifer einen Namen, der dem der angegriffenen Bank sehr ähnlich ist. Nun verschickt der Angreifer massenhaft Email in denen sinngemäß folgendes steht:

> Lieber *Bank*--Kunde,
> wir hatten leider technische Probleme und daher müssen Sie sich leider erneut auf unserem Banking-Server registrieren. Klicken Sie bitte einfach auf die folgende URL:
> `https://banking.bamk.de/registierung/`

[4] Prada und Armani sind Markennamen von Kleidungsstücken.

Vielen Dank für Ihr Verständnis
Ihre *Bank*

Oftmals wird die Email auch als HTML verschickt, so daß auf dem Client-Rechner eine andere URL angezeigt wird als diejenige, zu der ein Klick auf die URL tatsächlich führt. Ein Beispiel:

```
<a href="http://angreifer.de"> http://www.opfer.de </a>
```

In HTML ist es der Wert des Attributs `href`, der angibt wohin ein Klick des Benutzer denn führt. Der Inhalt des Elements `<a>` (zwischen Start tag und End tag) ist das, was dem Benutzer auf der HTML-Seite angezeigt wird. Auch der Einsatz von Kodierungstechniken für URLs kann dazu führen, die URL für den Menschen schwer überprüfbar zu machen.

Deshalb empfiehlt es sich, keine URLs aus verdächtigen Emails direkt anzuklicken. Statt dessen sollte man die Adresse seiner Bank in die Favoriten-Liste des Browsers eintragen und immer nur über die Liste anklicken. Außerdem schadet es auch nicht, hin und wieder das SSL-Zertifikat zu überprüfen.

1.7 Gefährliche Benutzereingaben

Viele Rechner sind gefährdet, wenn sie Benutzereingaben übernehmen und diese dann ungeprüft an weitere Programme übergeben. Böswillige Benutzer könnten dann Eingaben machen, die die Programmierer so nicht erwartet haben. Wenn diese Eingaben dann verarbeitet werden, können Sicherheitsprobleme entstehen. In Abschnitt 1.7.1 geht es darum, daß manche Benutzereingaben später in HTML-Seiten dynamisch eingefügt werden. Diese Seiten werden dann von anderen Benutzern geladen, d. h. vom Browser des anderen Benutzers, und interpretiert. Dabei können dann unerwünschte Effekte auftreten.

Abschnitt 1.7.2 diskutiert das Problem, wenn Benutzereingaben direkt an eine Datenbank weitergeleitet werden. In vielen Web-Anwendungen werden SQL-Anweisungen direkt anhand der Benutzereingaben zusammengestellt und dann ausgeführt. Mit passenden Benutzereingaben kann ein Angreifer unbefugt die Datenbank manipulieren.

1.7.1 HTML Code

In HTML-Seiten können aktive Inhalte integriert sein. Diese aktiven Inhalte bergen einige Sicherheitsrisiken; sie sind z. B. in Kurs *(01866) Sicherheit im Internet I* beschrieben. Das können sich Angreifer zunutze machen, wenn sie Eingaben erzeugen dürfen, die dann später in HTML-Seiten eingebettet sind.

Ein typisches Beispiel hierfür sind Gästebücher auf persönlichen Web-Sites. Dort kann jedermann in einem HTML-Formular Einträge für das Gästebuch verfassen. Schaut sich ein anderer Benutzer den Inhalt des Gästebuches an, dann werden die Eingaben der anderen Gäste in eine HTML-Seite eingebettet und dem aktuellen Besucher angezeigt. Hat der Benutzer in ein Formular beispielsweise eingegeben:

```
Lieber Homepage Betreiber,

ich finde deine Seiten total toll und werde sie
allen meinen Freunden weiterempfehlen.

Viele Gruesse
Der Kritiker
```

Daraus kann dann bei Abruf der Seite der folgende HTML-Code generiert werden:

```
<html>
<head>
... der typische HTML Kopf ...
</head>
<body>
<h1>Grosses Gaestebuch von toller Hecht</h1>
Der Benutzer Kritiker schrieb am 1.4.2005 ueber
diese Seiten:
<pre>
Lieber Homepage Betreiber,

ich finde deine Seiten total toll und werde sie
allen meinen Freunden weiterempfehlen.

Viele Gruesse
Der Kritiker
</pre>
... hier kommen weitere Kritiken oder was auch immer ...
</body>
```

Hat ein vorheriger Besucher in der Eingabemaske für seinen Eintrag nun HTML-Elemente eingetippt, dann werden diese HTML-Elemente vom Browser des nachfolgenden Benutzers interpretiert und angezeigt. Ein böswilliger Benutzer könnte also `<script>`-, `<applet>`- oder ähnliche HTML-Elemente eintippen, die dann ausgeführt werden und dabei möglicherweise Schäden verursachen.

1.7.2 SQL injection

In vielen E-commerce Anwendungen müssen Daten aus Datenbanken abgerufen werden. Meldet sich beispielsweise ein Benutzer an, dann muß die Benutzer-Kennung in einer Datenbank nachgeschlagen werden. Eventuell wird sogar zusätzlich noch ein Paßwort überprüft. Aber auch an anderen Stellen sind Datenbankenzugriffe erforderlich. Beispiele hierfür sind: Warenauswahl aus einem Katalog, Verfügbarkeitsüberprüfungen, etc.

Structured Query Language (SQL)

Als Sprache für die Datenbankabfragen (engl. **query**) wird meistens **Structured Query Language (SQL)** benutzt. Weitere Details zum Thema Datenbanken und SQL werden in Kurs *(01665) Datenbanksysteme* vorgestellt. In

SQL kann man neben Anfragen (engl. **query**) aber auch Manipulationen an einer Datenbank vornehmen. Eine einfache SQL-Anfrage sieht beispielsweise so aus:

```
select * from user where kennung = 'stefan';
```

Diese Anweisung sucht in der Tabelle `user` nach allen Zeilen, die in der Spalte `kennung` die Zeichenkette `stefan` enthalten. Die Werte aller Spalten dieser Zeilen werden dann ausgegeben.

Wichtig für das Verständnis von SQL injection sind die folgenden drei Konzepte von SQL:

1. Zeichenketten (engl. **strings**) werden in einfache Anführungszeichen gesetzt.
2. Mehrere SQL-Anweisungen werden durch ein Semikolon getrennt.
3. Kommentare in SQL-Anweisungen beginnen mit `--`. Der Rest der Zeile wird dann vom SQL-Interpreter ignoriert.

In Internet-Anwendungen möchte man dem Benutzer erlauben, die Zeichenketten selbst einzugeben. Die Anwendung baut dann aus den Eingaben die SQL-Anweisung zusammen. Steht die Benutzereingabe beispielsweise in einer Variablen `eingabe`, dann kann man in einem Java-Programm durch die folgende Zeile eine SQL-Anweisung konstruieren:

```
String anw = new String("select * from user where
            kennung = '" + eingabe + "';");
```

Der Java-Operator + konkateniert Strings, die in Java in doppelte Anführungszeichen gesetzt sind. Hat die Variable `eingabe` den Wert `stefan`, dann ensteht in der Variablen `anw` die oben bereits gezeigte SQL-Anweisung. Um diese Anweisung tatsächlich auszuführen, braucht es in **Java Database Connectivity (JDBC)** noch eine Verbindung zur Datenbank und ein Statement-Objekt. Der Java-Code sieht dann so aus:

Java Database Connectivity (JDBC)

```
import java.sql.*;

// Sei getConnection eine Methode, die
// die Verbindung zur Datenbank aufbaut.

Connection conn = getConnection();
Statement  stat = conn.getStatement();
String     anw = new String( /* siehe oben */ );

ResultSet rs = stat.executeQuery(anw);

// Nun noch die Ergebnismenge verarbeiten
```

Wenn der Benutzer in die Variable `eingabe` aber nun folgendes eingibt, entsteht eine neue SQL-Anweisung. Aus der Eingabe:

```
'; drop table user;--
```

wird diese SQL-Anweisung:

```
select * from user where kennung = ''; drop table user; --';
```

Diese Anweisung ist eine Sequenz aus zwei Anweisungen. Die erste durchsucht die Tabelle `user` und die zweite Anweisung (`drop table user;`) löscht die Tabelle `user` aus der Datenbank.

Die Ursache dieses Angriffs liegt darin, daß der Benutzer ein einfaches Hochkomma eingeben konnte und damit die ursprüngliche SQL-Anweisung „vorzeitig" beenden kann. Der Rest der Eingabe wird als neue SQL-Anweisung interpretiert. Man kann dort alle SQL-Anweisungen eingeben und damit beliebige Manipulationen an der Datenbank vornehmen. Dazu ist es allerdings erforderlich, die Namen der Tabellen und der Attribute (Spalten) zu kennen. Kennt man sie nicht, dann müsste ein Angreifer die Namen raten. Voraussichtlich wird er dabei erst einmal falsche Namen ausprobieren. Je nach Datenbanksystem werden dabei verschieden ausführliche Fehlermeldungen erzeugt. Diese Fehlermeldungen enthalten dann möglicherweise die bisher unbekannten Tabellen- oder Attribut-Namen.

Gegenmaßnahmen

Um diese Angriffe zu verhindern, müssen Anwendungen den Benutzereingaben *immer* mißtrauen. Jede Eingabe muß von der Anwendung überprüft werden. Hierzu können zwei Strategien eingesetzt werden:

1. Durchsuche die Eingabe nach „gefährlichen" Zeichen und ersetze sie durch ungefährliche Alternativen.

2. Lasse in der Eingabe nur bestimmte Zeichen (oder reguläre Ausdrücke) zu.

Die erste Variante hat den Nachteil, daß man beim Entwurf der Prüfung Zeichen für ungefährlich hält, die sich später aber evtl. als gefährlich herausstellen. Dann wäre die Überprüfung unvollständig und muß geändert werden. Die zweite Variante ist sicherer, da sie die ungefährlichen Eingaben beschreibt und alles Abweichende abweist.

Java-Programmierer, die den Datenbankzugriff mit JDBC implementieren, können an Stelle eines *Statement*-Objekts auch ein sogenanntes *PreparedStatement* (vorbereitete Anweisung) benutzen. In einem PreparedStatement dürfen nur an bestimmten Stellen die Benutzereingaben einkopiert werden. Das Datenbanksystem kann bei einem PreparedStatement vorab die günstigste Ausführungsreihenfolge planen. Benutzt man das PreparedStatement dann mehrmals (immer mit anderen Parametern), so spart die Datenbank bei den Folgeaufrufen die Planung und kann die Anfrage so schneller beantworten. Außerdem kann das PreparedStatement-Objekt prüfen, ob die Benutzereingaben die Struktur der Anweisung verändern oder nicht. Der Java-Code sieht dann so aus:

```
import java.sql.*;

// Sei getConnection eine Methode, die
// die Verbindung zur Datenbank aufbaut.

Connection conn = getConnection();

String anw = new String("SELECT * FROM user" +
             " WHERE kennung = ?" );

PreparedStatement pstat = conn.prepareStatement(anw);

// Der String eingabe enthalte die Benutzereingabe
// Setze den 1. Parameter (? im Anweisungs-String)
// auf den Wert der Variablen eingabe
   pstat.setString(1, eingabe);

ResultSet rs = pstat.executeQuery();

// Und wieder die Ergebnismenge verarbeiten
```

Der String für das PreparedStatement enthält an den Stellen, an denen später noch Werte gesetzt werden sollen, nur ein Fragezeichen. Die set-Methoden des PreparedStatement-Objekts können nun den Wert einer Variablen an Stelle eines Fragezeichens einfügen. Und setString kann beispielsweise prüfen, daß der Wert tatsächlich ein „normaler“ String ist, der kein einfaches Anführungszeichen enthält. Weitere Details zu JDBC finden Sie beispielsweise in [HC05].

1.8 Spezielle „Hacker“ Tools

Es existieren verschiedene Hilfsprogramme, mit denen die Sicherheit eines Systems getestet werden kann. Hierbei kann man drei wesentliche Klassen unterscheiden:

1. **Host Scanner** untersuchen einen einzelnen Rechner und testen beispielsweise die Rechte-Vergabe sowie Stärke der Paßwörter. — Host Scanner

2. **Network Scanner** untersuchen ein Netz und konzentrieren sich auf die Kommunikation zwischen Rechnern. — Network Scanner

3. **Intrusion Scanner** versuchen einen Einbruch in einen Rechner oder ein Netz zu erkennen. Sie werden später im Kurs behandelt (siehe Abschnitt 3.3). — Intrusion Scanner

In den beiden folgenden Unterabschnitten werden die beiden erstgenannten Klassen vorgestellt.

1.8.1 Host Scanner

Problembereiche

Ein Host Scanner untersucht die Konfiguration eines Rechners auf Schwachstellen. Das war früher deutlich wichtiger als heute, weil einige Betriebssystemhersteller ihre Systeme im Auslieferungszustand unsicher vorkonfiguriert hatten. So wurden beispielsweise viele Dienste aktiviert (der Benutzer könnte sie ja brauchen), obwohl die meisten Benutzer diese Dienste nicht benutzen wollten und gar nicht wussten, daß dieser Dienst aktiviert war. Angreifer fanden darüber aber einen Startpunkt. Heute sind die meisten Hersteller vorsichtiger und liefern ihre Systeme in einer sicheren Grundkonfiguration. Der Administrator muß Änderungen selbst vornehmen und sollte dabei dann natürlich wissen was er macht. Host Scanner untersuchen die folgenden Punkte:

Zugriffsrechte von Dateien: Ein typisches Problem das hier auftreten kann sind Leserechte für zu viele Benutzer. Ein Wörterbuchangriff (siehe Abschnitt 1.4) benötigt Zugriff auf die Paßwort-Datei. Falls diese Datei mit Leserechten für alle Benutzer versehen ist, kann dieser Angriff Erfolg haben.

Besitzer von Dateien: Falls eine Datei einem falschen Besitzer oder einer falschen Gruppe zugeordnet ist, kann ein Angreifer diese Datei unberechtigt lesen oder verändern. Ausführbare Dateien mit dem SUID-Bit laufen mit den Rechten des Besitzers der Datei, nicht mit den Rechten desjenigen, der das Programm startet (siehe auch Kurs *(01801) Betriebssysteme und Rechnernetze*). So kann ein normaler Benutzer auf die Paßwort-Datei zugreifen (z. B. um sein Paßwort zu ändern), ohne selbst die Zugriffsrechte zu besitzen. Statt dessen ist das Programm, das die Paßwort-Datei liest und verändert, mit entsprechenden Rechten ausgestattet. Das Programm gehört dem Administrator und läuft daher mit dessen Rechten.

Ein Angreifer könnte nun versuchen, sein Programm mit gesetztem SUID-Bit dem Benutzer `root` zuzuordnen. Dieses Programm enthält eine Schadens-Funktion und würde, egal von welchem Benutzer es gestartet wird, immer die Zugriffsrechte von `root` haben und könnte die Schäden dann auch immer anrichten.

Unbefugte Modifikationen: Ein Host Scanner kann auch die Prüfung der Integrität von Daten vornehmen. Mit Hilfe von digitalen Signaturen können Veränderungen entdeckt und evtl. auch rückgängig gemacht werden.

Inhalte von Konfigurationsdateien: Falls auf dem System spezielle Systemsoftware, beispielsweise ein Datenbanksystem installiert wurde, so enthält die zugehörige Konfiguration häufig einige Probleme. Es können Standard-Benutzer mit bekannten Standard-Paßwörtern vorkonfiguriert sein. Diese Konfiguration sollte auf einem produktiven System natürlich geändert sein. Bei Aktualisierungen (engl. **update**) der Software sollte die Konfiguration jedesmal mitgeprüft werden.

Versionen der Software: In vielen Programmen wurden Sicherheitsprobleme festgestellt. Diese wurden in aktuelleren Versionen behoben. Ein Host Scanner kann prüfen, ob Programme mit bekannten Schwachstellen auf dem System installiert sind.

Im folgenden werden die Host Scanner *COPS* für Linux und *MBSA* für MS Windows vorgestellt.

COPS (Computerized Oracle and Password System) wurde von Dan Farmer und Eugene Spafford [FS90] entwickelt und ist eine Sammlung von Shell-Skripten oder Perl-Programmen. Jedes dieser Programme untersucht einen anderen potentiellen Problembereich. Findet das Skript ein mögliches Problem, so wird der Benutzer darauf aufmerksam gemacht. Das Skript versucht *nicht*, das Problem zu lösen. Zusammen mit *COPS* wird ein Hilfsprogramm *CARP (COPS Analysis and Report Program)* geliefert. Dieses Programm unterstützt den Benutzer bei der Auswertung der Ergebnisse von COPS. Insbesondere kann man mehrere Analyseergebnisse von *COPS* übersichtlich in Tabellenform vergleichen. Die folgenden Problembereiche werden von *COPS* geprüft:

COPS

Problembereiche

Zugriffsrechte auf Dateien, Verzeichnisse und Geräte (`/dev/*`),

Inhalte, Format und Sicherheit der Paßwort- und Gruppendateien,

Programme und Dateien im Verzeichnis `/etc/rc*`,

Existenz und Zugriffsrechte von Programmen die das **SUID**-bit (Set User ID) gesetzt haben, die dem Benutzer `root` gehören und Shell-Skripts sind,

SUID

Schreibrechte in den persönlichen Verzeichnissen (engl. **home directories**) der Benutzer,

Konfiguration des Dienstes `ftp`, insb. ob anonymes ftp erlaubt ist,

CERT Hinweise und ob möglicherweise einige Programme in einer alten Version installiert sind, so daß sie besser ausgetauscht werden sollten und das

Kuang Expertensystem. Es enthält eine Reihe von Regeln mit denen es testet, ob das System möglicherweise verletzlich ist.

Grundsätzlich muß man davon ausgehen, daß das Programm zwar mögliche Schwachstellen entdeckt, aber ein *COPS*-Lauf ohne Warnmeldungen bedeutet umgekehrt *nicht*, daß das System sicher ist. Wenn man *COPS* benutzen möchte, geht man wie folgt vor:

Vorgehensweise

1. Man sucht mit einer Internet-Suchmaschine nach den Quellen von *COPS* und sucht sich eine vertrauenswürdige Quelle, von der man die Quelltexte lädt.

2. Man packt die Quellen aus und liest die mitgelieferte Dokumentation.

3. Mit Hilfe des Programms `./reconfig` ändert man die Skripte, so daß die Pfadnamen in den Skripten an das eigene System angepaßt sind. Beispielsweise wird das Hilfsprogramm *awk* benutzt und *reconfig* trägt den aktuellen Pfad (z. B. `/usr/bin/awk`) in die einzelnen *COPS*-Skripte ein.

4. Mit dem Kommando `make` erzeugt man einige Hilfsprogramme, die aus den *COPS*-Skripten aufgerufen werden.

5. Anschließend kann man *COPS* starten.

Ausgabe

Das Ergebnis von *COPS* sieht dann möglicherweise so aus:

```
ATTENTION:
Security Report for Sam Aug 31 20:40:52 CEST 2002
from host james, COPS v. Version 1.04+^M

**** root.chk ****
**** dev.chk ****
**** is_able.chk ****
**** rc.chk ****
**** cron.chk ****
**** group.chk ****
**** home.chk ****
Warning!  User wohlfeil's home directory /home/wohlfeil is mode 0722!
**** passwd.chk ****
**** user.chk ****
**** misc.chk ****
**** ftp.chk ****
ftp-Warning! Incorrect permissions on "ls" in /usr/local/ftp/bin!
ftp-Warning! Incorrect permissions on "passwd" in /usr/local/ftp/etc!
ftp-Warning! Incorrect permissions on "group" in /usr/local/ftp/etc!
**** kuang ****
**** bug.chk ****
```

In diesem Beispiel wurde das persönliche Verzeichnis des Benutzers `wohlfeil` absichtlich mit Schreibrechten für jedermann versehen. Diese Ausgabe kann *COPS* auch in eine Datei schreiben. Diese steht standardmäßig in einem Verzeichnis mit dem Namen des Rechners. Als Dateiname erzeugt *COPS* das aktuelle Datum des Prüfung, beispielsweise `2002_Aug_31`.

Das Programm *CARP* erstellt nun eine Tabelle mit den aktuellsten Ergebnissen der Überprüfung von mehreren Rechnern. Sie kann einmal als ASCII Ausgabe erstellt werden und sieht dann so aus:

```
COPS warning summary

hostname      rep date     crn dev ftp grp hme is pass msc pwd rc root usr kng
================================================================================
james         2002_Aug_31  |   |   | 2 |   | 1 |   | ? |   |   |   |   |   |   |
butler        2002_Aug_30  |   |   | 2 |   |   |   | ? |   |   |   |   |   |   |
```

Dabei haben die Ziffern in den Zellen der Tabelle folgende Bedeutungen:

- 0 Ein Angreifer könnte sofort `root` Zugriff erhalten
- 1 Ein anderes ernstes Problem
- 2 Eine Meldung von *COPS*, nicht weiter dramatisch
- ? Eine unbekannte/fehlende Meldung

Mit einem weiteren Hilfsprogramm kann man die Tabelle auch direkt in PostScript umwandeln, so daß die Ergebnisse graphisch dargestellt werden. Abbildung 1.6 zeigt die Ausgabe der oben gezeigten Tabelle.

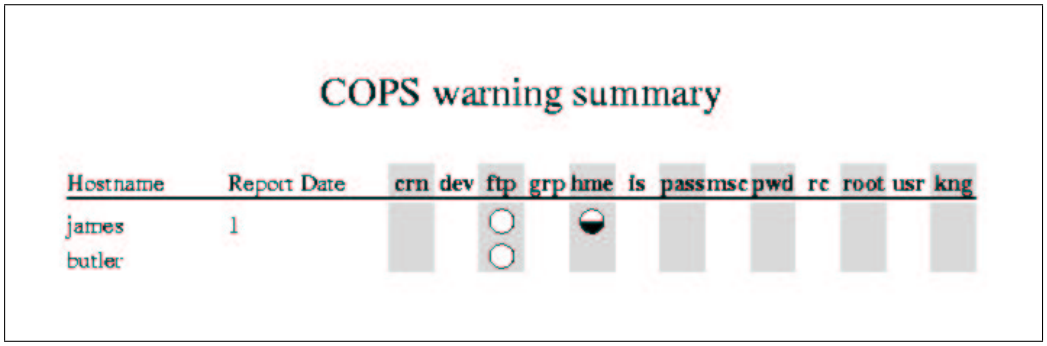


Abbildung 1.6: Ausgabe des Programms *COPS*

Microsoft Baseline Security Analyser (MBSA): Dieses Programm kann man kostenlos von den Webseiten der Firma *Microsoft* herunterladen. Es dient im wesentlichen dazu, den Sicherheitsstatus von Rechnern mit einem Betriebssystem der Windows-Familie zu prüfen. Konkret geht es um:

- Überprüfung der Konfiguration eines Rechners, beispielsweise auf eingeschaltete Dienste, Laufwerks-Freigaben, Sicherheit der Paßwörter gegen brute force Angriffe, automatische Anmeldung, usw.
- Überprüfung ob auch alle Sicherheitsaktualisierungen (engl. **security updates**) für das Betriebssystem installiert sind. Dazu muß sich das Programm natürlich mit einem Server bei *Microsoft* verbinden.
- Überprüfung weiterer Programme von Microsoft (z. B. der Web-Server *Internet Information Server*, der Datenbank-Server *MS SQL Server*, usw.), falls sie installiert sind.

Hinweise zu Microsoft Baseline Security Analyser finden Sie im Internet unter der URL:

`http://technet.microsoft.com/de-de/security/cc184923.aspx`

In Abbildung 1.7 ist eine Beispielausgabe von *MBSA* dargestellt. Das einzige schwerwiegende Sicherheitsproblem auf einem Windows 7 Laptop ist eine FAT-Partition. Der Grund ist, daß auf FAT-Partitionen keine Dateizugriffsrechte verwaltet werden können. Eine solche Partition eignet also nur als Austauschpartition. Hat man mehrere Betriebssysteme auf einem Rechner installiert, so kann man sich immer sicher sein, daß jedes Betriebssystem diese FAT-Partition lesen und schreiben kann. Außerdem hat MBSA festgestellt, daß die Paßwörter niemals ablaufen. Die Möglichkeit, daß Benutzer regelmäßig ihre Paßwörter ändern müssen, ist deaktiviert.

Neben der Überprüfung des Rechners auf dem es läuft kann der MBSA auch andere Rechner im Netz überprüfen. Das leitet direkt zum nächsten Abschnitt über.

1.8.2 Network Scanner

Ein Network Scanner untersucht zusätzlich, welche Netzdienste auf einem Rechner installiert und freigeschaltet sind. Diese Netzdienste, wie beispielswei-

Netzdienste