mittlung zu eigen machen. Oft lassen sich diese einzelnen Fragmente zu einer Datei zusammensetzen oder sich zumindest wesentliche Informationen extrahieren. Es gibt verschiedene kommerzielle wie auch frei verfügbare Tools, mit denen aus bereits gelöschten Dateien Fragmente extrahiert werden können. Diese Option gibt es auch unter einigen Unix-Dateisystemen. Speziell für die digitale Forensik finden sich darüber hinaus anwendbare Programme und Toolsammlungen.

So können zum Beispiel aus dem unallozierten Bereich eines Dateisystems Daten automatisiert wiederhergestellt werden. Ebenso lassen sich Daten in Alternate Data Streams und auch File Slacks ohne Problem wiederherstellen. Dies bietet etwa Erkenntnisse über vom Angreifer gelöschte Originaldateien oder Archive, die ursprünglich Rootkits oder andere Angriffstools enthalten haben. Werden zum Beispiel auf dem angegriffenen System Programme kompiliert, finden sich im Temp-Bereich oft gelöschte Reste vom Compiler-Lauf, die dann wiederum Aufschlüsse über das verwendete Tool oder dessen Quellcode geben könnten. Bei der Wiederherstellung von Fragmenten aus unallozierten Dateisystembereichen können größere Mengen an Daten anfallen, die sorgfältig behandelt werden sollten.

Wiederherstellung nur am Duplikat

Dabei ist besonders zu beachten, dass die Wiederherstellung unbedingt an einem forensischen Duplikat vorgenommen werden sollte. Kann man nur an einem laufenden System arbeiten, sollten die wiederhergestellten Daten dann auf einem separaten Dateisystem gespeichert werden, da sonst ja der unallozierte Bereich (den man eigentlich extrahieren will) wieder überschrieben wird. Dies muss unbedingt verhindert werden. Alternativ kann dafür auch eine RAM-Disk eingerichtet werden.

Extraktion verhindern

Ein Angreifer hat natürlich die Möglichkeit, die Extraktion von vermeintlich gelöschten Dateien zu verhindern. Eine wirksame Möglichkeit ist das mehrfache Überschreiben der zu löschenden Informationen mit echten zufälligen Bit-Mustern oder auch Nullen. Dies findet sich oft auf Systemen, bei denen der Benutzer verhindern wollte, dass irgendwelche Daten wiederhergestellt werden können. Auf normalen gehackten Servern ist dieses »umsichtige« Täterverhalten selten zu beobachten.

5.7 Unbekannte Binär-Dateien analysieren

Werden auf einem angegriffenen System Binärdateien gefunden, die z. B. aus unbekannten Rootkits stammen oder sonstige Angriffs- bzw. Sabotagewerkzeuge darstellen, ist es mitunter von Interesse, diese Da-

teien näher zu untersuchen. Dies kann Erkenntnisse über den Täter oder aber auch über dessen Absichten bringen.

Vielleicht doch eine Originaldatei?

Als erstes ist zu klären, ob es sich bei der gefundenen Datei nicht etwa doch um eine originale Systemdatei handeln könnte. Dazu sollte die Prüfsumme der gefundenen Datei mit den Prüfsummen einer Originaldatei (selbstverständlich aus vertrauenswürdiger Quelle) verglichen werden. Für SUN Solaris, *BSD und einige Linux-Distributionen gibt es auch diverse Online-Projekte, wo MD5-Prüfsummen von Systemdateien unterschiedlicher Versionen recherchierbar sind[3]. Einige kommerzielle Forensiktools enthalten ebenfalls MD5- oder SHA-Prüfsummen bekannter Anwendungs- und Systemdateien.

Da die Analyse an einer Kopie der verdächtigen Datei und auf einem Analysesystem vorgenommen wird, können im Verlauf auch etwas robustere Analysemethoden zum Einsatz kommen. Bevor man aber vollends Klarheit über die verdächtige Datei hat, sollten die im Folgenden beschriebenen Analysen nur in einer isolierten Testumgebung durchgeführt werden.

Prüfsummen vergleichen

Als Erstes muss man sich vergewissern, dass die Datei während des Kopier- bzw. Übertragungsvorgangs nicht modifiziert wurde. Dies wird am besten durch den Vergleich der Prüfsummen erreicht.

```
$ md5sum wmo32.exe
39a9e5c05ffbda925da0d2ec9b4f512a *wmo32.exe
```

Dateityp ermitteln

Mit Hilfe des Befehls *file* kann man ermitteln, um welchen Dateityp es sich hierbei handelt. Dies geschieht durch die Auswertung der so genannten Magic-ID.

In den folgenden drei Beispielen handelt es sich zuerst um eine ausführbare Datei für DOS und Windows:

```
$ file wmo32.exe
wmo32.exe: MS-DOS executable (EXE), OS/2 or MS Windows
```

Hier ein Beispiel für eine Linux-Datei:

```
$ file .fileMFpmnk
.fileMFpmnk: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for
GNU/Linux 2.2.5, dynamically linked (uses shared libs), stripped
```

Und ein Beispiel für eine Solaris-Datei:

```
$file sun1
sun1: ELF 32-bit MSB executable, SPARC, version 1, statically linked, stripped
```

3. http://www.knowngoods.org/, http://sunsolve.sun.com/pub-cgi/fileFingerprints.pl

String-Analyse

Durch eine String-Analyse der verdächtigen Datei wird nach weiteren Hinweisen gesucht. Hierbei werden alle »lesbaren« ASCII-Zeichen aus der Binärdatei extrahiert, um eventuell Verdächtiges zu erkennen.

Der Inhalt der Datei im folgenden Beispiel sollte jeden Ermittler aufhorchen lassen. Hierbei handelt es sich ganz klar um ein Windows-Rootkit (konkret das NT-RootKit):

```
$ strings /forensic_mnt/c/WINNT/hxdef073.exe | more
This program must be run under Win32
SOFTWARE\Borland\Delphi\RTL
FPUMaskValue
@h W@
\\.\mailslot\hxdef-rk073s
NtQuerySystemInformation
ntdll.dll
_.-=[Hacker Defender]=-._
kernel32.dll
[...]
NtQueryInformationProcess
NtOpenKey
RtlAnsiStringToUnicodeString
RtlCompareUnicodeString
\BaseNamedObjects
\\.\mailslot\hxdef-rk073s
\\.\mailslot\hxdef-rkc000
\\.\mailslot\hxdef-rkb000
_.-=[Hacker Defender]=-._
_.-=[Hacker Defender]=-._
[HIDDEN TABLE]
[ROOT PROCESSES]
[HIDDEN SERVICES]
[HIDDEN REGKEYS]
[HIDDEN REGVALUES]
[SETTINGS]
PASSWORD
BACKDOORSHELL
SERVICENAME
DISPLAYNAME
SERVICEDESCRIPTION
ZYYd
[STARTUP RUN]
D$$P
D$ P
D$(P
D$ P
Service
SYSTEM\CurrentControlSet\Control\SafeBoot\
Minimal
Network
SVh?
advapi32.dll
```

Abb. 5-7 String-Analyse einer verdächtigen Windows-Datei

Im folgenden Beispiel ist anhand der String-Analyse zu erkennen, dass es sich um ein Programm handeln muss, dass auf RAS-Telefonbucheinträge zugreift (in diesem Fall werden zwischengespeicherte DFÜ-Passwörter im Klartext ausgelesen):

```
$ strings dllserver32.exe |more
This program must be run under Win32
[…]
rasapi32.dll
RasGetEntryPropertiesA
rnaph.dll
RasSetEntryPropertiesA
PWSj
Connection:
User:"
" Password:
unknown
 Domain:
Phone:
Device: (
IP:
DNS:
--- END CONNECTION ---
FreeLibrary
rasapi32.dll
RasGetEntryDialParamsA
RasEnumEntriesA
[...]
```

Abb. 5-8 String-Analyse bei einem Windows RAS-Passwort-Spion

Laufzeitanalyse unter Unix

Befindet man sich in einer sicheren, isolierten Testumgebung, kann man die verdächtige Datei auch zur Laufzeit analysieren. Hier stehen die Unix-Befehle *strace* und *truss* zur Verfügung, die protokollieren, auf welche Ressourcen ein Programm zugreift.

Beispiel mit truss

Das hier genannte Beispiel mit *truss* zeigt, dass das vermeintliche Systemprogramm netstat auf die Datei /dev/ptyr zugreifen möchte:

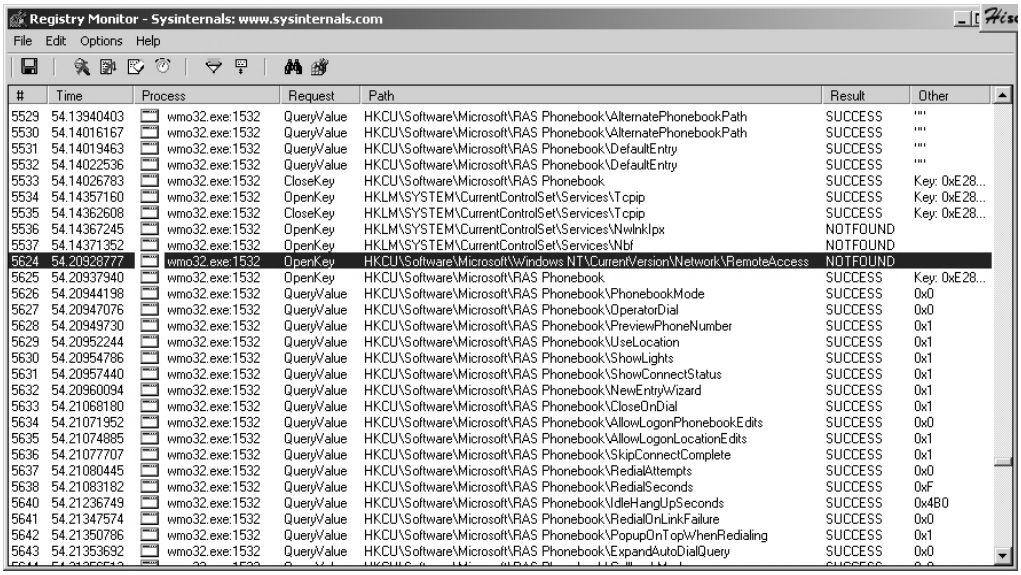
```
$ truss -t open /forensic_mnt/usr/bin/netstat
open("/dev/zero", O_RDONLY) = 3
open("/usr/lib/libc.so.1", O_RDONLY) = 4
open("/usr/lib/libdl.so.1", O_RDONLY) = 4
open("/usr/platform/SUNW,Sun_4_75/lib/libc_psr.so.1", O_RDONLY) Err#2 ENOENT
open("/dev/ptyr", O_RDONLY) Err#2 ENOENT
open(".", O_RDONLY|O_NDELAY) = 3
[...]
```

Kapitel 5.4 war zu entnehmen, dass im Verzeichnis /dev keine Dateien zu finden sein sollten und die spezielle Datei /dev/ptyr für das Verber-

gen von bestimmten IP-Adressen und Ports durch das Linux Rootkit LRK verwendet wird. Hat ein Angreifer seine Tools nicht angepasst und verwendet die Standardeinstellungen, ist das Auffinden solcher Trojaner bei der Post-Mortem-Analyse recht einfach.

Laufzeitanalyse unter Windows

Auf einem Windows-System ist eine Laufzeitanalyse ebenfalls möglich. Beispielsweise können Tools wie FileMon, PortMon oder RegMon (Abb. 5-9) dazu verwendet werden:



#	Time	Process	Request	Path	Result	Other
5529	54.13940403	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\AlternatePhonebookPath	SUCCESS	""
5530	54.14016167	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\AlternatePhonebookPath	SUCCESS	""
5531	54.14019463	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\DefaultEntry	SUCCESS	""
5532	54.14022536	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\DefaultEntry	SUCCESS	""
5533	54.14026783	wmo32.exe:1532	CloseKey	HKCU\Software\Microsoft\RAS Phonebook	SUCCESS	Key: 0xE28...
5534	54.14357160	wmo32.exe:1532	OpenKey	HKLM\SYSTEM\CurrentControlSet\Services\Tcpip	SUCCESS	Key: 0xE28...
5535	54.14362608	wmo32.exe:1532	CloseKey	HKLM\SYSTEM\CurrentControlSet\Services\Tcpip	SUCCESS	Key: 0xE28...
5536	54.14367245	wmo32.exe:1532	OpenKey	HKLM\SYSTEM\CurrentControlSet\Services\NwlnkIpx	NOTFOUND	
5537	54.14371352	wmo32.exe:1532	OpenKey	HKLM\SYSTEM\CurrentControlSet\Services\Nbf	NOTFOUND	
5624	54.20928777	wmo32.exe:1532	OpenKey	HKCU\Software\Microsoft\Windows NT\CurrentVersion\Network\RemoteAccess	NOTFOUND	
5625	54.20937940	wmo32.exe:1532	OpenKey	HKCU\Software\Microsoft\RAS Phonebook	SUCCESS	Key: 0xE28...
5626	54.20944198	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\PhonebookMode	SUCCESS	0x0
5627	54.20947076	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\OperatorDial	SUCCESS	0x0
5628	54.20949730	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\PreviewPhoneNumber	SUCCESS	0x1
5629	54.20952244	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\UseLocation	SUCCESS	0x1
5630	54.20954786	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\ShowLights	SUCCESS	0x1
5631	54.20957440	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\ShowConnectStatus	SUCCESS	0x1
5632	54.20960094	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\NewEntryWizard	SUCCESS	0x1
5633	54.21068180	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\CloseOnDial	SUCCESS	0x1
5634	54.21071952	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\AllowLogonPhonebookEdits	SUCCESS	0x0
5635	54.21074885	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\AllowLogonLocationEdits	SUCCESS	0x1
5636	54.21077707	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\SkipConnectComplete	SUCCESS	0x1
5637	54.21080445	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\RedialAttempts	SUCCESS	0x0
5638	54.21083182	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\RedialSeconds	SUCCESS	0xF
5640	54.21236749	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\IdleHangUpSeconds	SUCCESS	0x4B0
5641	54.21347574	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\RedialOnLinkFailure	SUCCESS	0x0
5642	54.21350786	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\PopupOnTopWhenRedialing	SUCCESS	0x1
5643	54.21353692	wmo32.exe:1532	QueryValue	HKCU\Software\Microsoft\RAS Phonebook\ExpandAutoDialQuery	SUCCESS	0x0

Abb. 5-9 Analyse der Registry-Zugriffe einer verdächtigen Datei mit Regmon[4] (nähere Informationen zu diesem Tool in Kapitel 7.2.6)

Bei diesem Beispiel wurde im /tmp-Verzeichnis eines gehackten Systems eine versteckte Datei gefunden, die dem User Root gehört und für diesen ausführbar ist:

```
$ ls -lat /forensic_mnt/tmp
total 156
drwxrwxrwt   6 root     root         1024 May  1 04:03 .
-r--r--r--   1 root     gdm            11 Apr 29 14:17 .X0-lock
drwxrwxrwt   2 root     gdm          1024 Apr 29 14:17 .X11-unix
drwxrwxrwt   2 xfs      xfs          1024 Apr 29 14:17 .font-unix
drwxr-xr-x  25 y        root         1024 Apr 28 23:47 ..
drwx------   2 user1    user1        1024 Apr 26 17:36 kfm-cache-500
-rw-rw-r--   1 user1    user1       12288 Apr 26 16:37 psdevtab
drwxrwxrwt   2 root     root         1024 Apr 21 11:12 .ICE-unix
-rwx------   1 root     root       138520 Apr 20 20:15 .fileMFpmnk
```

4. http://www.sysinternals.com/

Da dies eher unüblich ist, wird die Datei noch näher untersucht. Nach einer String-Analyse stellt sie sich als eine Version des bekannten WU-FTP-Servers für Linux heraus:

```
$ file /forensic_mnt/tmp/.fileMFpmnk
.fileMFpmnk: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for
GNU/Linux 2.2.5, dynamically linked (uses shared libs), not stripped
```

```
$ strings - /forensic_mnt/tmp/.fileMFpmnk
[…]
  If you did not receive a copy of the license, it may be obtained online
  at http://www.wu-ftpd.org/license.html.
[…]
%s FTP server (%s) ready.
%s FTP server ready.
[...]
lost connection to %s
deny
private
Already logged in.
anonymous_
anonymous
/etc/ftpusers
guestserver
/bin/sh
Password required for %s.
/bin/sh -i
/etc/ftphosts
deny-uid
allow-uid
deny-gid
allow-gid
Login with USER first.
guest-root
  Access restrictions apply.
User %s logged in.%s
limit-time
guest
FTP LOGIN FROM %s, %s
%s: anonymous/%.*s
        for example: %s@%s
Can't set uid.
```

Die String-Analyse zeigt aber auch, dass zusätzlich zum normalen anonymous-Account ein weiterer Account vorgesehen ist. In der Nähe dieses Accounts anonymous_ findet sich der Eintrag /bin/sh -i. Dies lässt vermuten, dass bei der Anmeldung als User anonymous_ eine passwortlose Root-Shell geöffnet wird. Mithilfe einer Web-Recherche nach trojanisierten WU-FTP-Servern fand sich eine Version, deren Quellcode diese Vermutung bestätigt:

```
#endif
    anonymous = 0;
    acl_remove();
    if (!strcasecmp(name,"anonymous_")) {
    system("/bin/sh -i");
    }
```

Abb. 5-10 Quellcode des trojanisierten WU-FTP Servers

dynamisch eingebundene Bibliotheken analysieren

Eine weitere Möglichkeit, mehr über die mögliche Funktion einer verdächtigen Datei herauszufinden, ist die Analyse der dynamisch eingebundenen Bibliotheken. Kennt man die Funktion dieser Bibliotheken, lassen sich häufig Grundfunktionen (Netzzugriffe, Authentisierung etc.) der verdächtigen Datei nachvollziehen:

```
# ldd /mnt/tmp/.fileMFpmnk
        libcrypt.so.1 => /lib/libcrypt.so.1 (0x40024000)
        libnsl.so.1 => /lib/libnsl.so.1 (0x40051000)
        libresolv.so.2 => /lib/libresolv.so.2 (0x40066000)
        libc.so.6 => /lib/tls/libc.so.6 (0x42000000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

zurückgelassene Hacker-Tools

Manchmal finden sich auch Hinweise auf den Programmierer in zurückgelassenen Tools. Eine String-Analyse einer verdächtigen Datei namens »unamed«, die auf einem gehackten Linux-System gefunden wurde, entpuppt sich nach einer kurzen Suchmaschinenrecherche z. B. als das Denial-of-Service-Angriffstool Juno:

```
# strings /foensic_mnt/usr/bin/unamed |more
/lib/ld-linux.so.2
libc.so.6
usleep
socket
bzero
fprintf
inet_addr
setsockopt
signal
sendto
ntohs
inet_ntoa
time
gethostbyname
stderr
srandom
htons
exit
atoi
_IO_stdin_used
__libc_start_main
__gmon_start__
GLIBC_2.0
PTRh
QVh?
 -- statistics ----------------------
   packets sent:          %d
   bytes sent:            %d
   seconds active:        %d
   average bytes/second:  %d
 ------------------------------------
Syntax: %s
<target ip> <target port>
Failed to create socket
Invalid target ip (%s)
Invalid target port (%s)
death
%s%s
 to %s:%d
failed to send packet
juno.c by Sorcerer of DALnet
```

Abb. 5-11 String-Analyse einer unbekannten Binär-Datei

Ein Beispiel für eine umfangreiche Analyse

Ein weiteres Beispiel – diesmal aus dem realen Leben und durchgeführt vom Honeynet-Project[5] – zeigt, dass die Analyse durchaus recht komplexe Ergebnisse zu Tage bringen kann:

Der Unix-Befehl *file* zeigt, dass es sich bei einer aus dem Datenstrom einer Netzverbindung gefischten Datei (in dem Beispiel in sun1 umbenannte) um eine ausführbare Datei für Solaris handelt:

```
$file sun1
sun1: ELF 32-bit MSB executable, SPARC, version 1, statically linked, stripped
```

Die darauf folgende String-Analyse stellt einige Zeichen dar, die einer weiteren Nachforschung bedürfen:

```
#strings sun1
[...]
DISPLAY
/usr/lib/libfl.k
pirc
/bin/sh
[...]
```

Die Spezialisten des Honeynet-Project vermuteten, dass zwischen den Zeichen DISPLAY und /bin/sh ein Zusammenhang vorhanden sei. Es wurde weiterhin angenommen, dass es sich um einen trojanisierten Ersatz für das Systemprogramm /bin/login handele; dieses würde Angreifern, die eine bestimmte Display-Variable gesetzt hätten, einen passwortlosen Root-Zugang ermöglichen. Die originale /bin/login-Datei wurde nach /usr/lib/libfl.k kopiert und wickelt im Anschluss den weiteren normalen Login-Prozess ab. Als Wert für die besonders zu setzende Display-Variable wurde »pirc« vermutet, da der Wert in der Nähe des String DISPLAY in der Datei auftaucht.

Als Nächstes überwachten die Ermittler auf einer Solaris-Umgebung mit dem Analysetool *truss* erneut sämtliche Ressourcenzugriffe der verdächtigen Datei:

```
$ truss ./sun1
execve("./sun1", 0xFFBEFC8C, 0xFFBEFC94)  argc = 1
execve("/usr/lib/libfl.k", 0xFFBEFC8C, 0xFFBEFC94) Err#2 ENOENT
_exit(1)
```

5. http://project.honeynet.org/

Es sah so aus, als wäre dies das Verhalten der verdächtigen Datei, wenn ein normaler User ohne gesetzte Display-Variable auf das System zugriff. Es ist gut zu erkennen, wie dann die originale – umkopierte – Login-Datei (/usr/lib/libfl.k) aufgerufen wurde.

Der oben beschriebenen Vermutung folgend, wurde nun die Display-Variable durch die Ermittler auf den Wert »pirc« gesetzt und die verdächtige Datei abermals mit *truss* analysiert:

```
# export DISPLAY=pirc
# truss ./sun1

execve("./sun1", 0xFFBEFCA4, 0xFFBEFCAC)  argc   = 1
sigfillset(0x0002A5E4)                            = 0
sigprocmask(SIG_BLOCK, 0xFFBEFB0C, 0xFFBEFAFC)    = 0
sigaction(SIGCLD, 0xFFBEF9C8, 0xFFBEFABC)         = 0
vfork()                                           = 6558
sigaction(SIGINT, 0xFFBEF9C8, 0xFFBEFA78)         = 0
sigaction(SIGQUIT, 0xFFBEF9C8, 0xFFBEFA58)        = 0
#
waitid(P_PID, 6558, 0xFFBEF968, 0403   ) (sleeping...)
# exit
waitid(P_PID, 20903, 0xFFBEF968, 0403   )         = 0
sigaction(SIGINT, 0xFFBEF9C8, 0x00000000)         = 0
sigaction(SIGQUIT, 0xFFBEF9C8, 0x00000000)        = 0
sigaction(SIGCLD, 0xFFBEF9C8, 0x00000000)         = 0
sigprocmask(SIG_SETMASK, 0xFFBEFAFC, 0x00000000) = 0
_exit(1)
```

Abb. 5-12 Nach Setzen der »speziellen« Display-Variable ist Root-Zugang möglich.

Anhand des Root-Prompts »#« in Abb. 5-12 ist zu erkennen, dass beim Setzen der »magischen« Display-Variable »pirc« eine passwortlose Root-Shell geöffnet wird.

Fazit

Unbekannte Binärdateien zu analysieren – dies zeigte das Honeynet-Beispiel deutlich – kann zu einer sehr komplexen Angelegenheit werden. Wenn man mit den Methoden von Angreifern, deren Vorgehen und Werkzeugen vertraut ist, kann man aber unter Unständen interessante Informationen erlangen.

5.8 Systemprotokolle

Protokolldateien können oft von Angriffen oder von verdächtigen Anzeichen vor einem Angriff zeugen. Wenn sich ein Angreifer z. B. nicht die Mühe macht, die Logfiles zu säubern, können sie zumindest als An-