Rootkits: Techniken und Abwehr

Von Rootkits eingesetzte Techniken und Methoden zur ihrer Entdeckung und Abwehr

DFN Workshop 2003, 26. Februar 2003
Dipl. Inf. Andreas Bunten

Gliederung

- Was sind Rootkits?
- Klassen von Rootkits und Abwehrmaßnahmen
  - Austausch von Systembefehlen
  - Kernel Rootkits
  - Neuere Entwicklungen bei Kernel Rootkits
  - Rootkits unter Microsoft Windows
- Allgemeine Maßnahmen
- Zusammenfassung und Ausblick

Was sind Rootkits?

DFN CERT

Szenario:

- Auf einem kompromittiertem System erlangte der Angreifer die Rechte des Administrators
- Der Angreifer will das System unbemerkt für seine Zwecke missbrauchen

Ziel des Vortrags:

- Was ist einem Angreifer mit Rootkits möglich?
- Wie können Rootkits entdeckt oder abgewehrt werden?

Austausch von Befehlen

DFN CERT

Vorgehensweise des Angreifers:

- Ersetzen von Befehlen wie `ls`, `ps` und `netstat`
- Neue Versionen unterdrücken manche Ausgaben

Möglichkeiten durch Austausch von Befehlen:

- Verstecken von Dateien, Prozessen und Verbindungen des Angreifers
- Bereitstellung von Hintertüren ins System

Den Ausgaben der Befehle kann nicht mehr vertraut werden.

Austausch von Befehlen

Beispiel: Rootkit *torn* tauscht Befehl `ls`

- Schlüsselwörter aus `/usr/src/.puta/.1file` werden eingelesen
- Funktionalität wie das Original, aber:
  Ausgabe enthält keine Zeile mit Schlüsselwort.

Austausch von Befehlen

Entdeckung des Rootkits:

- Test von Checksummen: Tripwire / AIDE / rpm
  – Hoher Aufwand bei Pflege
  + Schadensabschätzung möglich
- Manuelle Suche mit 'frischen' oder alternativen Programmen
  – Für jedes Rootkit ex. andere Merkmale
  + Für praktisch jedes Rootkit ex. Merkmale
- Automatisierte Suche, z.B.: Chkrootkit

Kernel-Rootkits

DFN CERT

Kernelmodul: Zur Laufzeit ladbare Teile des Kerns

Systemaufrufe: Schnittstelle zu Betriebsmitteln des Kernels

Befehle benutzen bei Ausführung Systemaufrufe:

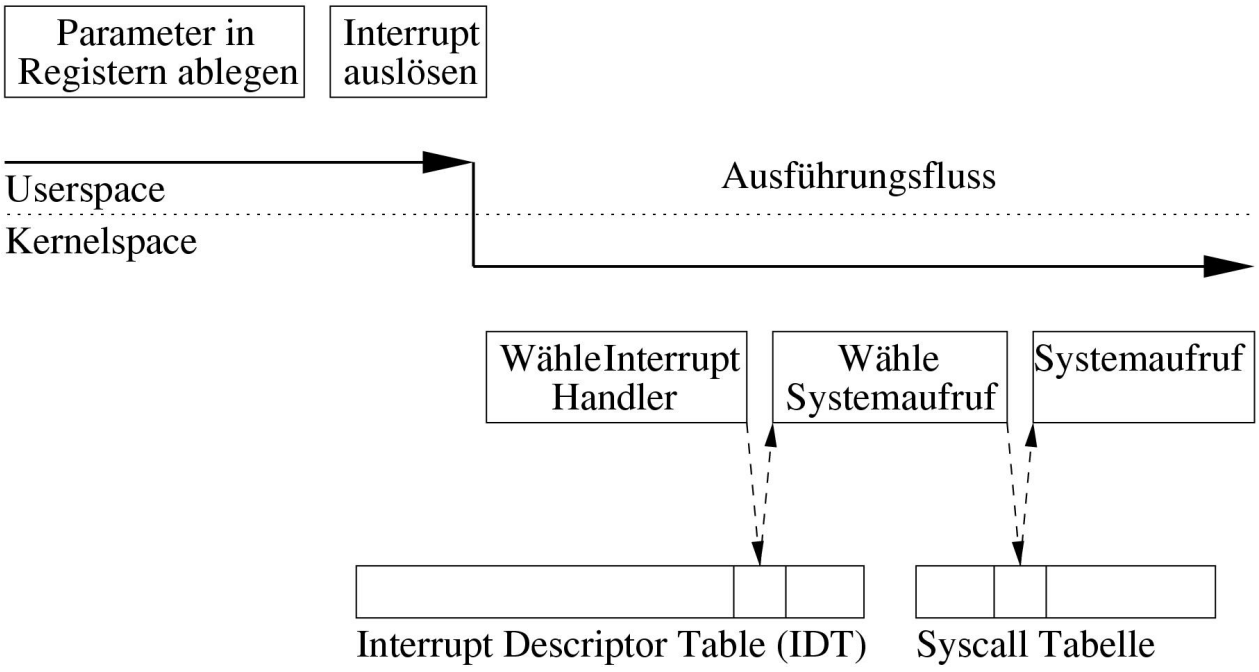
Kommandoebene:	`ls /tmp`

Systemaufrufe in Assemblerbene:	`...`
	`open(/tmp, O_RDONLY)`
	`getdents( ... )`
	`write( ... )`
	`...`

Kernel-Rootkits

DFN CERT

Ablauf eines Systemaufrufs:



Dipl. Inf. Andreas Bunten - © 2002 DFN-CERT GmbH

Kernel-Rootkits

DFN CERT

Vorgehensweise des Angreifers:

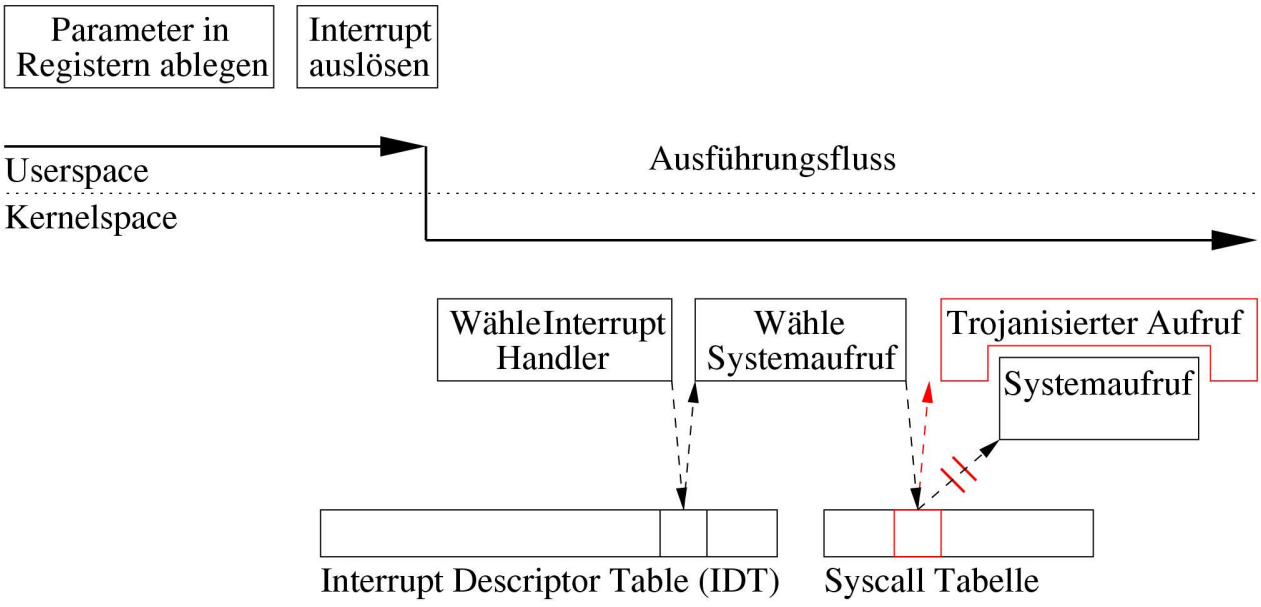
- Rootkit als Modul in den Kernel laden
- Austausch von Systemaufrufen durch Manipulation der Syscall Tabelle
- Neue Systemaufrufe filtern Rückgabewerte der Originale
- Benutzer Kommandos erhalten verfälschte Informationen – es ist kein Austausch nötig.

Das System selber wird kompromittiert. Bisherige Abwehrmaßnahmen sind nur begrenzt anwendbar.

Kernel-Rootkits

DFN CERT

Ablauf eines trojanisierten Systemaufrufs:



Dipl. Inf. Andreas Bunten - © 2002 DFN-CERT GmbH

Kernel-Rootkits

DFN CERT

Beispiel: Kernel-Rootkit *adore*

- Es werden u.a. `stat()`, `getdents()` und `write()` ausgetauscht
- Durch Userspace Programm `ava` werden Schlüsselwörter angegeben
- In den Ausgaben der Systemaufrufe werden die Schlüsselwörter unterdrückt
- Der Befehl `ls` benutzt manipulierte Systemaufrufe

Kernel-Rootkits

DFN CERT

Neue Möglichkeiten der Kernel-Rootkits:

- Executable Redirection
- Direkte Manipulation des TCP/IP-Stacks
- Manipulation beim Auslesen von Speicherseiten
- Zufälliges Einfügen von Fehlern bei Ausführung von Systemaufrufen

Kernel Rootkits manipulieren dazu meist nur wenige Systemaufrufe.

Kernel-Rootkits

DFN CERT

Entdeckung von Kernel-Rootkits:

- Suche nach unbekannten Modulen
  – leicht zu umgehen
- Kontrolle kritischer Strukturen im Kernel
  – Referenzwerte müssen vorher gesichert werden
  + Viele Rootkits manipulieren wenige Strukturen
- Suche nach speziellen Merkmalen
  – Für jedes Rootkit ex. andere Merkmale
  + Für praktisch jedes Rootkit ex. Merkmale

Kernel-Rootkits

DFN CERT

Beispiel: Kontrolle kritischer Strukturen

```
linux:/home/tools/KSTAT24/2.4.16 # ./kstat -s 0
sys_fork      0xf880c7a0 WARNING! should be at 0xc01058fc
sys_write     0xf880ca30 WARNING! should be at 0xc013193c
sys_open      0xf880dbd0 WARNING! should be at 0xc01312c8
sys_close     0xf880cb80 WARNING! should be at 0xc0131450
sys_oldstat   0xf880cff0 WARNING! should be at 0xc0137dd4
sys_kill      0xf880c900 WARNING! should be at 0xc011e1c8
sys_mkdir     0xf880ccb0 WARNING! should be at 0xc013bfc8
sys_oldlstat 0xf880d1e0 WARNING! should be at 0xc0137eb4
sys_stat      0xf880d3d0 WARNING! should be at 0xc0137e44
...
```

Dipl. Inf. Andreas Bunten - © 2002 DFN-CERT GmbH

Kernel-Rootkits

DFN CERT

Beispiel: Suche nach speziellen Merkmalen

```
linux:/home/tools # ./chkproc -v -v
CWD   153: /
EXE   153: /usr/sbin/sshd

linux:/home/tools # ls -al /proc | grep 153

linux:/home/tools # cd /proc/153
linux:/proc/153 # cat cmdline ; echo
/usr/sbin/sshd
```


Kernel-Rootkits

DFN CERT

Abwehrmaßnahmen:

- Deaktivieren von Kernelmodulen
  – Erschwert Administration
  + Installation des Rootkits deutlich schwerer
- Anti-Rootkit Kernelmodule
  + Installation des Rootkits schwerer

Neue Kernel-Rootkits

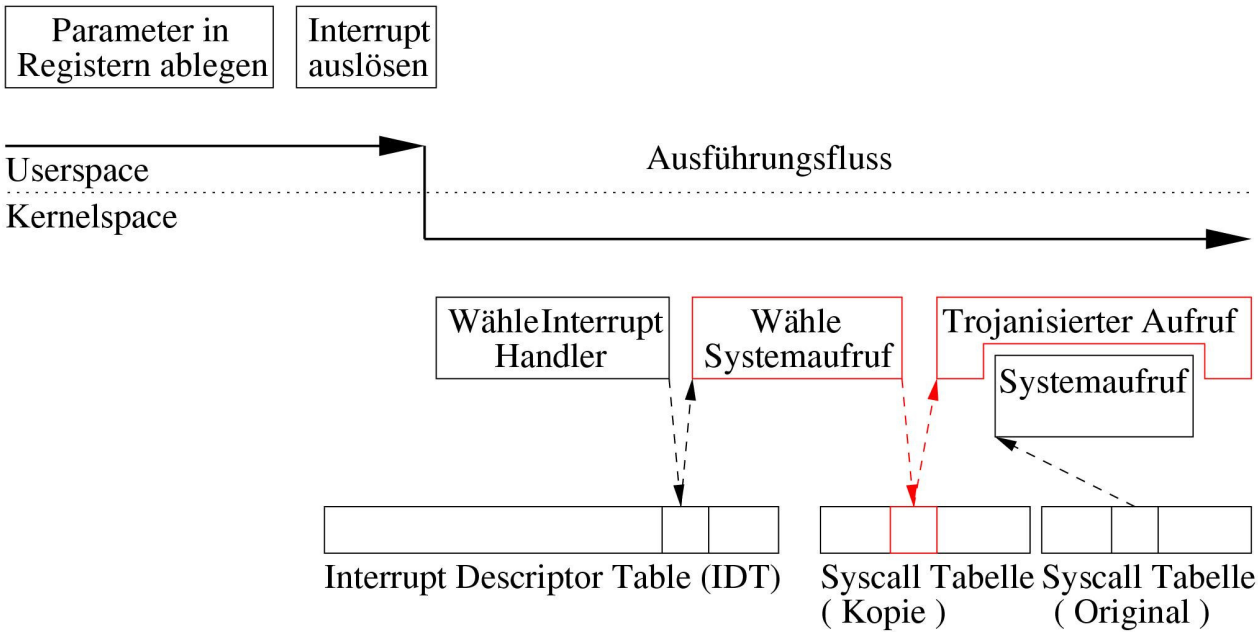
Neue Entwicklungen:

- Manipulation verschiedener Ressourcen des Kernels
- Umgehung der Modulunterstützung
- Einfügen trojanisierten Programmcodes in den Fluss des Systemaufrufs.

Bisherige Abwehrmaßnahmen sind nicht mehr hilfreich.

Neue Kernel-Rootkits

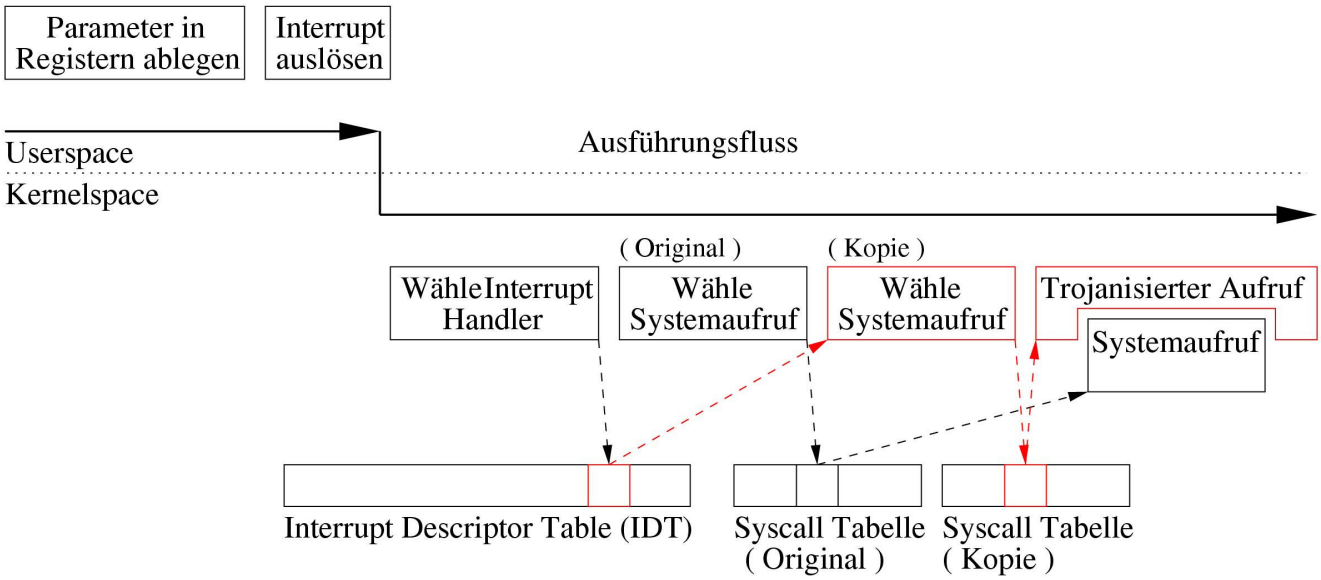
Manipulation verschiedener Ressourcen:



Dipl. Inf. Andreas Bunten - © 2002 DFN-CERT GmbH

Neue Kernel-Rootkits

Manipulation verschiedener Ressourcen:



Neue Kernel-Rootkits

Umgehung der Modulunterstützung:

- Zugriff über `/dev/kmem`
- Heuristische Suche nach Funktion `kmalloc()`
- Syscall Tabelle anhand Interrupt Handler finden
- Einen Aufruf durch `kmalloc()` ersetzen und so Speicher im Kernel reservieren
- Schreiben des trojanisierten Programmcodes
- Restauration der Syscall Tabelle und Eintragen des trojanisierten Systemaufrufs

Neue Kernel-Rootkits

Beispiel: Kernel-Rootkit *sk*

- Installation ohne Modulunterstützung
- Verwendung einer Kopie der Syscall Tabelle
- Es werden 24 Systemaufrufe ausgetauscht
- Übersetztes Binary lauffähig unter Linux 2.2.x und 2.4.x
- Versteckt Verbindungen, Dateien und Prozesse
- Backdoor über das Netzwerk und Persistenz bereits im Rootkit implementiert.

Neue Kernel-Rootkits

Entdeckung neuer Kernel-Rootkits:

- Laufzeitmessung:
  – Ungenauigkeiten bei Messung und Modul nötig
  + Sehr allgemeine Methode
- Suche nach speziellen Merkmalen
  – Für jedes Rootkit ex. andere Merkmale
  + Für praktisch jedes Rootkit ex. Merkmale

Neue Kernel-Rootkits

DFN CERT

Beispiel: Laufzeitmessung

```
linux:/home/tools # ./patchfinder -c referenz_2.4.16
(...)
    test name       | current | clear | diff   | status
-----------------------------------------------------------
open_file           |    7110|    1442|   5668| ALERT!
stat_file           |    7050|    1255|   5795| ALERT!
read_file           |     608|     608|      0|  ok
open_kmem           |    7124|    1510|   5614| ALERT!
readdir_root        |    6497|    2750|   3747| ALERT!
readdir_proc        |   14422|    2401|  12021| ALERT!
(...)
```


Neue Kernel-Rootkits

DFN CERT

Beispiel: Suche nach speziellen Merkmalen

```
linux:/sbin # ls -il init*
 68269 -rwxr-xr-x 1 root root 429356 Jan 3 18:32 init
linux:/sbin # md5sum init
225e82f77d74dc8326335381d47f87a9  init

linux:/sbin # mv init Zinit
linux:/sbin # ls -il Zinit
 69665 -rwxr-xr-x 1 root root  28984 Jan 3 18:32 Zinit
linux:/sbin # md5sum Zinit
89e5bcb0ef33614bb8c3a45c59d5e471  Zinit
```

Neue Kernel-Rootkits

Abwehrmaßnahmen:

- Deaktivieren des Schreibzugriffs auf `/dev/kmem`
  - – Verzicht auf manche HW-nahe Programme
  - – Unterstützung von ACLs notwendig
  - + Installation des Rootkits deutlich erschwert

Windows Rootkits

Ähnlichkeiten Windows und UNIX:

- Programmcode kann zur Laufzeit in den Kern gebracht werden
- Native API stellt quasi Systemaufrufe zur Verfügung

Rootkits unter Windows sind Kernel-Rootkits unter UNIX vergleichbar.

Windows Rootkits

Beispiel: *NT Rootkit*

- Versteckt Dateien, Prozesse, Verbindungen und Registry-Einträge
- Executable Redirection
- Konsolen-Sniffer
- Gezielter Absturz des Systems möglich
- Backdoor über das Netzwerk bereits integriert.

Windows Rootkits

Entdeckung von Windows Rootkits:

- Anti-Viren Software
  – Entdeckt bisher wenige Rootkits
  – Wird leicht von Rootkit manipuliert
  + Ist bereits auf vielen Systemen vorhanden
- Suche nach speziellen Merkmalen
  – Für jedes Rootkit ex. andere Merkmale
  + Für praktisch jedes Rootkit ex. Merkmale

Windows Rootkits

Abwehrmaßnahmen:

- Anti-Rootkit Treiber
  - – Sind teilweise leicht zu umgehen
  - – Evtl. Stabilitätsprobleme des Systems
  - + Installation des Rootkits etwas erschwert

Allgemeine Maßnahmen

Unabhängig vom System:

- Kontrolle des Netzwerkes
  - Zentrale Logstruktur (*Loghost*)
  - Statistiken zum Netzwerkverkehr
- Analyse des verdächtigen Systems offline
  - Untersuchung in kontrollierter Umgebung
  - Forensische Analyse

Alle bisher vorgestellten Maßnahmen verlassen sich auf das kompromittierte System.

Zusammenfassung

- Es kommen nicht nur aktuelle Rootkits zum Einsatz – es muss mit allem gerechnet werden!
- Allgemeine Konsistenz-Tests sind besser als Suche nach speziellen Merkmalen
- Zentrales Logging und Kontrolle des Netzerkes erleichtert die Entdeckung
- Notfallpläne und praktische Erfahrung erleichtert die Reaktion.

Ausblick

- Entwicklung am schnellsten auf freien UNIX-artigen Systemen
- Rootkits sind einfacher zu bedienen und bringen oft schon Backdoor mit
- Entdeckung wird deutlich schwerer, wenn keine zentralen Ressourcen manipuliert werden
- Platz für Forschung:
  - Allgemeine Konsistenz-Tests
  - Robuste Erkennung von Rootkits

Diskussion

Wir sehen uns gerne Ihre Rootkits an.

bunten@dfn-cert.de