

UNIX Dateirechte

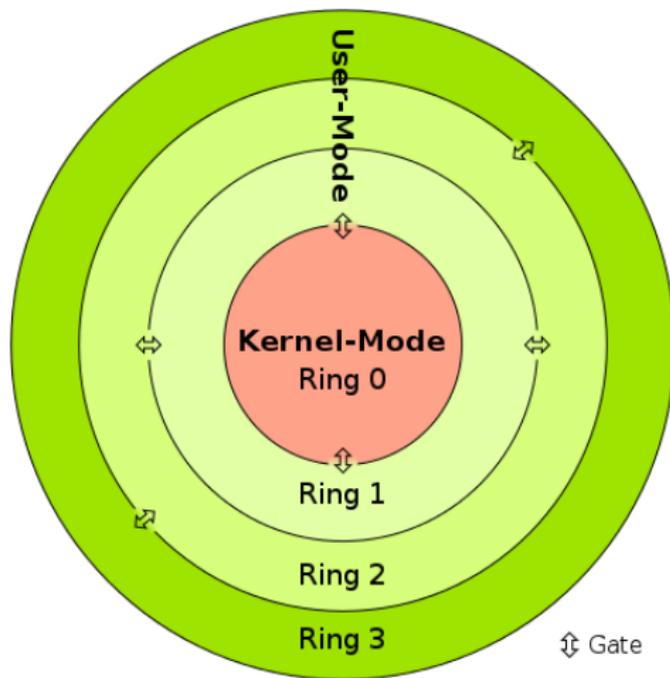
Michael Hartmann

Linux User Group Augsburg

7. Oktober 2015

Wie funktioniert eigentlich
Sicherheit auf Computern?

Sicherheitskonzept



CPU-Ringe

Funktionsweise:

- Ring (Domain): Sicherheitsstufe auf der CPU
- innerer Ring: darf alles
- äußere Ringe: eingeschränkter Befehlssatz
- Linux benutzt nur zwei Ringe (Kernel- und Benutzer-Modus)

Vorteile:

- Ringe ermöglichen Betriebssystem Sicherheitsmechanismen
- Einschränken von Hardware-Zugriff
- Abschotten von unterschiedlichen Prozessen
- ermöglicht Virtualisierung (in der Theorie...)

CPU-Ringe

Funktionsweise:

- Ring (Domain): Sicherheitsstufe auf der CPU
- innerer Ring: darf alles
- äußere Ringe: eingeschränkter Befehlssatz
- Linux benutzt nur zwei Ringe (Kernel- und Benutzer-Modus)

Vorteile:

- Ringe ermöglichen Betriebssystem Sicherheitsmechanismen
- Einschränken von Hardware-Zugriff
- Abschotten von unterschiedlichen Prozessen
- ermöglicht Virtualisierung (in der Theorie...)

Wie greift dann ein Prozess auf Hardware zu?

Systemaufrufe

Systemaufrufe!

Systemaufrufe

Ablauf:

- ➊ Prozess legt an einer bestimmten Stelle Zahlenwert / Daten ab
- ➋ Prozess löst Software-Interrupt ab
- ➌ Prozess wird unterbrochen, CPU wechselt in Ring 0
- ➍ Funktion im Kernel erledigt Aufgabe
- ➎ CPU wechselt wieder in Ring 3, Prozess wird weiter ausgeführt

Vorteile:

- Prozess hat zu keinem Zeitpunkt Zugriff auf Hardware
- Kernel kann entscheiden, ob Prozess Zugriff auf *Ressource* bekommt

Systemaufrufe

Ablauf:

- ➊ Prozess legt an einer bestimmten Stelle Zahlenwert / Daten ab
- ➋ Prozess löst Software-Interrupt ab
- ➌ Prozess wird unterbrochen, CPU wechselt in Ring 0
- ➍ Funktion im Kernel erledigt Aufgabe
- ➎ CPU wechselt wieder in Ring 3, Prozess wird weiter ausgeführt

Vorteile:

- Prozess hat zu keinem Zeitpunkt Zugriff auf Hardware
- Kernel kann entscheiden, ob Prozess Zugriff auf *Ressource* bekommt

Was sind denn solche
Systemaufrufe?

Systemaufrufe

Auswahl von Linux-Systemaufrufen:

- open
- read
- write
- close

- kill
- chdir
- chmod

- sendmsg
- recvmsg

Wer darf was?

Wer:

- **u**ser (Eigentümer)
- **g**roup (Gruppe)
- **o**ther (alle anderen)

Was:

- **r**ead (lesen)
- **w**rite (schreiben)
- **e**xecute (ausführen)
- setuid, setgid, sticky bit

Wer darf was?

Wer:

- **u**ser (Eigentümer)
- **g**roup (Gruppe)
- **o**ther (alle anderen)

Was:

- **r**ead (lesen)
- **w**rite (schreiben)
- **e**xecute (ausführen)
- **s**etuid, **s**etgid, **s**ticky bit

Wer darf was?

Wer:

- **u**ser (Eigentümer)
- **g**roup (Gruppe)
- **o**ther (alle anderen)

Was:

- **r**ead (lesen)
- **w**rite (schreiben)
- **e**xecute (ausführen)
- setuid, setgid, sticky bit

chmod

	Berechtigung	Zahl
Oktal:	lesen	4
	schreiben	2
	ausführen	1

Ändern von Dateirechten:

- `chmod u=rw datei`
- `chmod u+rwx datei`
- `chmod g-rx datei`
- `chmod o+r datei`
- `chmod a+r datei`
- `chmod 700 datei`

Grundlegende Rechte

Lesen:

- Dateien: Benutzer darf Datei lesen
- Verzeichnisse: Benutzer darf Inhalt eines Verzeichnisses auslesen

Schreiben:

- Dateien: Benutzer darf Datei schreiben
- Verzeichnisse: Benutzer darf Dateien erstellen, umbenennen, löschen und deren Dateirechte ändern

Ausführen:

- Dateien: Benutzer darf Programm ausführen
- Verzeichnisse: Benutzer darf in das Verzeichnis wechseln

Grundlegende Rechte

Lesen:

- Dateien: Benutzer darf Datei lesen
- Verzeichnisse: Benutzer darf Inhalt eines Verzeichnisses auslesen

Schreiben:

- Dateien: Benutzer darf Datei schreiben
- Verzeichnisse: Benutzer darf Dateien erstellen, umbenennen, löschen und deren Dateirechte ändern

Ausführen:

- Dateien: Benutzer darf Programm ausführen
- Verzeichnisse: Benutzer darf in das Verzeichnis wechseln

Grundlegende Rechte

Lesen:

- Dateien: Benutzer darf Datei lesen
- Verzeichnisse: Benutzer darf Inhalt eines Verzeichnisses auslesen

Schreiben:

- Dateien: Benutzer darf Datei schreiben
- Verzeichnisse: Benutzer darf Dateien erstellen, umbenennen, löschen und deren Dateirechte ändern

Ausführen:

- Dateien: Benutzer darf Programm ausführen
- Verzeichnisse: Benutzer darf in das Verzeichnis wechseln

setuid/setgid

setuid (s):

- Dateien: ausführbares Programm wird mit Rechten des Besitzers der Datei ausgeführt
- Verzeichnisse: ignoriert (Linux)

setgid (s):

- Dateien: ausführbares Programm wird mit Rechten der Gruppe der Datei ausgeführt
- Verzeichnisse: neu erstellte Dateien erben die Gruppen-ID

sticky bit (t):

- Dateien: ausführbares Programm bleibt nach Beenden im Arbeitsspeicher (historisch)
- Verzeichnisse: nur noch der Besitzer einer Datei darf diese löschen (Beispiel: /tmp)

setuid/setgid

setuid (s):

- Dateien: ausführbares Programm wird mit Rechten des Besitzers der Datei ausgeführt
- Verzeichnisse: ignoriert (Linux)

setgid (s):

- Dateien: ausführbares Programm wird mit Rechten der Gruppe der Datei ausgeführt
- Verzeichnisse: neu erstellte Dateien erben die Gruppen-ID

sticky bit (t):

- Dateien: ausführbares Programm bleibt nach Beenden im Arbeitsspeicher (historisch)
- Verzeichnisse: nur noch der Besitzer einer Datei darf diese löschen (Beispiel: /tmp)

setuid/setgid

setuid (s):

- Dateien: ausführbares Programm wird mit Rechten des Besitzers der Datei ausgeführt
- Verzeichnisse: ignoriert (Linux)

setgid (s):

- Dateien: ausführbares Programm wird mit Rechten der Gruppe der Datei ausgeführt
- Verzeichnisse: neu erstellte Dateien erben die Gruppen-ID

sticky bit (t):

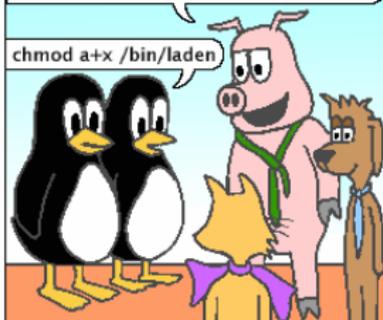
- Dateien: ausführbares Programm bleibt nach Beenden im Arbeitsspeicher (historisch)
- Verzeichnisse: nur noch der Besitzer einer Datei darf diese löschen (Beispiel: /tmp)

Hackles

Hackles

Hey guys, Percy has a 133t joke for us...

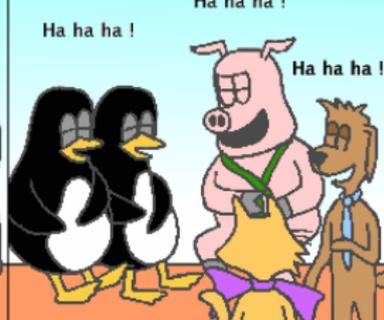
`chmod a+x /bin/laden`



Ha ha ha !

Ha ha ha !

Ha ha ha !



<http://hackles.org>

By Drake Emko & Jen Brodzik

I don't get it.



Copyright © 2001 Drake Emko & Jen Brodzik

<http://hackles.org/cgi-bin/archives.pl?request=65>

Vielen Dank für die
Aufmerksamkeit :)