Proseminar
Konzepte von Betriebssystem-Komponenten (KVBK)
Schwerpunkt Sicherheit

Superuser vs. dedizierte Capabilities (Trusted Solaris)

Vortragender: Marcel Beister (Beister@gmx.de)

Übersicht

- 1. Intro: Das Problem der Zugriffskontrolle
- 2. Die konventionelle Lösung: Superuser
- 3. Access Control Listen (ACLs)
- 4. Capabilities
- 5. Superuser vs. Capabilities
- 6. Capabilities unter Linux
- 7. Trusted Solaris: ACLs, Capabilities und administrative Rollen
- 8. Fazit
- 9. Quellen

1. Intro: Das Problem der Zugriffskontrolle

- Welche Objekte sind einem Subjekt zugänglich und in welcher Form?

- Objekte: Ressourcen wie etwa Dateien, Geräte, Netzwerke, Pipes, andere Prozesse,...
- Subjekte: Programme bzw. Benutzer
- Zugang: Art der Operation die auf die Objekte angewendet werden kann

1. Intro: Das Problem der Zugriffskontrolle

- Problem: Nicht alles läßt sich auf Dateizugriffe abbilden

- Beispiele:
  - Gerätetreiber oder Kernmodule
  - Direkter Hardware-Zugriff
  - TCP/UDP-Ports rohe IP-Pakete senden
  - Netzwerkkonfiguration
  - Scheduling
  - Signale an fremde Prozesse schicken
  - Zugriffe auf Pipes, Shared Memory, ...

2. Die konventionelle Lösung: Superuser

- klassische Unix-Ansatz:

  - Dateisystem: Zugriffsrechte für User, Group und Other

  - restliche Zugriffe: „root-sein-oder-nicht-sein“

- Nachteile:

  - Dateisystem: Einteilung grundsätzlich ziemlich grob

  - restliche Zugriffe: viele Programme müssen mit Root-Rechten gestartet werden (setuid / System Privilege Table)

2. Die konventionelle Lösung: Superuser

- Setuid: das Programm wird mit Root-Rechten ausgeführt

  => Problem: Sicherheitslücke ermöglicht Ausführung von belibigem Code mit Root-Rechten

- System Privilege Table: Liste mit Programmen, die spezielle Rechte haben müssen

  => Problem: Sicherheitslücke oder Austausch der Datei ermöglichen ebenfalls Ausführung mit Root-Rechten

3. Access Control Listen (ACLs)

- Weiterentwicklung des normalen User-Group-Other Systems
- Jedes Objekt hat Liste mit einzelnen Benutzer und Gruppen
  und die jeweiligen zulässigen Aktionen auf das Objekt
- Nur zur Beschränkung des Dateisystems geeignet
- Problem: Alle Programme die unter meinem Login laufen
  können alle meine Daten bearbeiten oder löschen
- Ansatz: neue ID für jede Arbeitsaufgabe
  => höherer Aufwand und längere ACLs
  => Laufzeitverhalten
- Weiteres Problem: Weitergabe meiner Rechte nur als Besitzer bzw.
  Root möglich

Übersicht

4. Capabilities

- Begriff Capabilitiy geht auf Dennis und Van Horn zurück („Programming Semantics for Multiprogrammed Computations“, 1966)
- Idee: Ein Programm benötigt um auf ein Objekt zugreifen und es bearbeiten zu können einen speziellen Schlüssel
- Subjekt erhält eine Referenz auf ein Objekt, welches alle Rechte die das Subjekt an dem Objekt besitzt, enthält
- Bei der Nutzung der Capability (Zugriff auf das Objekt) werden die Rechte überprüft und über Zugriff entschieden
- Ansatz ist dem Superuser-Ansatz engegengesetzt

4.1 Eigenschaften von Capabilities

- Einzig und allein der Besitz eines hinreichenden Capabilities gewährt den Zugriff, nicht etwa der Login/UID.
- Für ein Objekt kann es mehrer Capabilities geben, welche wiederum verschieden Zugriffsrechte gewähren können
- Capabilities können den Zugriff auf eine Sammlung von anderen Capabilities ermöglichen.
- Capabilities können übertragen bzw. vererbt werden
- Capabilities können kopiert werden
- Capabilities können wieder „entzogen“ bzw. annuliert werden

4.2. Methoden zur Absicherung der Capabilities

- Tagged Capabilities:

  Spezielle Hardware stellt sicher, dass nur als Capabilitiy markierte Daten auch als Capabilities verwendet werden können, d.h. auf ihnen können dagegen keine arithmetischen Operationen ausgeführt werden

- Partitioned Capabilities:

  Capabilities und Daten werden getrennt voneinander gespeichert in einem speziellem Capability Speicher, auf fast allen aktuellen CPUs emulierbar

- Sparse Capabilities:

  Capabilities sind "lang", d.h. nur ein kleiner Bruchteil aller möglichen Capabilities gültig, quasi Verschlüsselung

4.3 Funktionsweise Capability-Basierender Systeme

- jedes Programm hält einen Satz an Capabilities
- Hat ein Programm A ein Capabilitiy zur Kommunikation mit Programm B, dann können beide ihre Capabilities austauschen bzw. übertragen
- In den Besitz eines Capabilities kommt man normalerweise nur über irgend eine Art von Kommunikation
- unbegrenzte Anzahl an Capabilities würde das System mit der Zeit ausbremsen
  => nur eine begrenzte und kleine Anzahl an Capabilities erlaubt
- Oberstes Ziel: Der Satz an Capabilities muß so spezifisch und klein wie möglich sein

4.3 Funktionsweise Capability-Basierender Systeme

- Mit Capabilities läßt sich das „Principle of last privilege“ realisieren

  d.h. ein Programm darf gerade soviel im System machen, wie es ursprünglich vorgesehen ist, für weitere Aktionen besitzt es nicht die benötigten Capabilities bzw. Rechte

- Will ein Programm eine Aktion auf ein Objekt ausführen, dann muß es das Capability aufrufen und ihm die Art der Aktion übermitteln

  Soll beispielsweise unter UNIX der Systemcall read(fd, buf, sz) aufgerufen werden, dann wird die Operation statt auf einen normalen Filedeskriptor auf ein Capabilitiy angewendet also read(cap(), buf, sz)

5. Capabilities vs. Superuser

- 5.1. Zugriffsbeschränkung (Confinement)

  Situation: - Ein Programm arbeitet mit wichtigen geheimen Daten
  - Der Computer ist an ein Netzwerk angebunden
  - Senden von Daten durch Programm soll verhindert werden

  Superuser: Schutz durch Firewall nötig

  Capabilities: dem Programm wird keine Capability für den Netzwerkzugriff übergeben

5. Capabilities vs. Superuser

- 5.2. Priviligierte Programme

  Situation: - Programm zum Ändern des Passworts
  - benötigt Zugriff auf Passwort-Datei
  - Benutzer darf keinesfalls Zugriff auf Datei erhalten

  Superuser: Start des Programms als Root
  bzw. Aufnahme in System Privilege Table

  Capabilities: Programm erhält Capability zum (ausschließlichen) Zugriff auf Passwort-Datei

  weiteres Beispiel: Ping-Befehlt mit Raw-Ip-Paketen

5. Capabilities vs. Superuser

- 5.3. Zusammenarbeit

  Situation: - geheimes wertvolles Programm
  - Binary darf nicht vergeben werden
  - jemand will das Programm mit eigenen Daten testen
  - Daten sind evtl. auch nicht für andere bestimmt

  Superuser: keine Möglichkeit bekannt

  Capabilities: Programm kann laufen, ohne das man den Code zu Gesicht bekommt, gleichzeitig können die Rechte für das Programm so konfiguriert werden, daß es nicht möglich ist die fremden Daten einzusehen

5. Capabilities vs. Superuser

- 5.4. Die spezifische Rücknahme von Rechten

Situation: - Mitarbeiter verläßt die Firma bzw. wird in eine andere Abteilung versetzt

- die Rechte auf bestimmte Dokumente sollen zurückgenommen werden

Superuser: Login löschen bzw. die Rechte an den entsprechenden Objekten entfernen

Capabilities: Login löschen bzw. neuen Login erstellen
Problem: die Rechte sind nicht am Objekt sondern über das gesammte System verteilt

6. Capabilities unter Linux

- in der Kernelversion 2.1 wurde mit der Implementierung begonnen
- seit der Version 2.2 halbwegs praxistauglich
- Ziel: Abschaffung der Root-Abhängigkeit diverser Programme
  => Abschaffung diverser Sicherheitslücken
- Capabilities helfen die Root-Rechte in kleinere Abschnitte zu fassen
- Im 2.2er Kernel: 7 Capabilities aus dem Posix-Dokument 1003.1e
  sowie 20 weitere linuxspezifische Capabilities

6. Capabilities unter Linux

- einige wichtige Beispiele:

- CAP_CHOWN	Erlaubt das Ändern der Dateiinhaberrechte
- CAP_DAC_OVERRIDE	Setzt DAC Zugriffsbeschränkungen ausser Kraft
- CAP_DAC_READ_SEARCH	Setzt DAC Zugriffsbeschränkungen die mit Lesen und Suchen zusammenhängen ausser Kraft
- CAP_KILL	Erlaubt das Senden von Signalen an Prozesse
- CAP_SETGID	Erlaubt das Ändern der GID
- CAP_SETUID	Erlaubt das Ändern der UID
- CAP_SETPCAP	Erlaubt das Übertragen und Entfernen von Caps.
- CAP_LINUX_IMMUTABLE	Erlaubt das verändern geschützter Dateien
- CAP_NET_BIND_SERVICE	Erlaubt das Binden von Ports < 1024
- CAP_NET_RAW	Erlaubt den Gebrauch von Raw-Sockets

Übersicht

- 1. Intro: Das Problem der Zugriffskontrolle
- 2. Die konventionelle Lösung: Superuser
- 3. Access Control Listen (ACLs)
- 4. Capabilities
- 5. Superuser vs. Capabilities
- 6. Capabilities unter Linux
- 7. Trusted Solaris: ACLs, Capabilities und administrative Rollen
- 8. Fazit
- 9. Quellen

7. Trusted Solaris

- Trusted Solaris ist sicherheits-optimierte Version von Solaris

- Erweiterungen:
  - Abschaffung des Superusers
  - erweiterte Kontrolle über Dateizugriffe
  - Monitoring von Aktionen
  - Abfangen von Tastatureingaben verhindern
  - ...

7. Trusted Solaris

- 7.1 Mandatory Acces Controls (MAC)

  - Einteilung der Rechte in verschiedene Ebenen mittels Labels

  - hierarchische Sicherheitsstufen wie “public”, “private” oder “private-engeneering”

  - Benutzer können Labels nicht überwinden und bleiben in ihrer Ebene

  - vehindert das Daten unbeabsichtigt in falsche Hände geraten können

7. Trusted Solaris

- 7.2. Discretionary Access Controls (DAC)

  - DAC bedient sich der normalen Dateizugriffsrechte bzw. ACLs

  - Steuerung des Zugriffs auf Daten in Abhängigkeit der Benutzeridentität bzw. Gruppenmitgliedschaft

  - Root ist jedoch nicht von diesen Beschränkungen ausgenommen

  - DAC wird zusammen mit MAC für die Kontrolle des Dateisystems verwendet

7. Trusted Solaris

- 7.3. Privileges

  - Verwendung von Capabilities

  - Umsetzung des „least privilege“-Prinzips

  - im Detail Unterschiede in der Implementierung, grundsätzliche Funktionsweise ist die gleiche

  - Capabilities setzen Teile der DAC-Restriktionen ausser Kraft

  - Regelung aller Sicherheitsbeschränkungen die nicht auf Dateizugriff beziehen

7. Trusted Solaris

- 7.4. Role-Based Access Control (RBAC)

  - Aufteilung der administrativen Teile auf mehrer Administratoren
    => mehrere sich ergänzende Administratoren

  - normales einloggen mit anschließender Übernahme der administrativen Rolle(n)

  - alle administrativen Aktivitäten können überwacht, protokolliert und zurückverfolgt werden

  - standardmäßig folgende Rollen: Security administrator (secadmin)
    System administrator (admin)
    Operator (oper)

7. Trusted Solaris

- 7.5. Rights Profiles

  - Datenbank mit funktionsorientierten Rechte-Profilen

  - in hierarchischer Form miteinander kombinierbar

  - Profile können abgeändert und angepasst werden

  - schnelle und effektive Verteilung von Rechten und Restriktionen

8. Fazit

- hohe Sicherheit contra Effizienz?
- Vor- und Nachteile auf beiden Seiten
- Historische Gründe
- ...

9. Quellen

- [1] Linuxsecurity.com: Linux 2.4 - Next Generation Kernel Security
  http://www.linuxsecurity.com/feature_stories/kernel-24-security.html
- [2] Linux Capabilities FAQ 0.2
  ftp://ftp.guardian.no/pub/free/linux/capabilities/capfaq.txt
- [3] SecurityFocus HOME Infocus: Introduction to Linux Capabilities and ACL's
  http://www.securityfocus.com/infocus/1400
- [4] Skyhunter.com: Introduction To Capability Based Security
  http://www.skyhunter.com/marcs/capabilityIntro/index.html
- [5] EROS-OS.org: Essays on Capabilities and Security
  http://www.eros-os.org/essays/00Essays.html
- [6] Norman Hardy's Homepage
  http://www.cap-lore.com/index.html

9. Quellen

- [7] Trusted Solaris Operating System
  http://wwws.sun.com/software/solaris/trustedsolaris/index.html
- [8] Rule Set Based Access Control (RSBAC) for Linux – Overview
  http://www.rsbac.org
- [9] Ievlev Stanislav's Homepage
  http://linux.ru.net/~inger/
- [10] Betriebssystem HOWTO für UNIX/Linux von Prof. Jürgen Plate
  http://www.fs.ei.tum.de/admin/howto/unix/