Script **generated by TTT**

Title: Grundlagen_Betriebssysteme (01.02.2013)

Date: Fri Feb 01 08:30:04 CET 2013

Duration: 90:07 min

Pages: 21

Schutzmatrix

Das Konzept der Schutzmatrix wurde von B. Lampson eingeführt. Es verknüpft Schutzdomänen mit den zu schützenden Objekten.

Schutzdomänen

Schutzmonitor

Schutzmatrix ist typischerweise sehr groß und dünn besetzt ⇒ eine direkte Implementierung ist deshalb nicht sinnvoll.

Zugriffskontrollliste

Capability-Liste

Zusammenfassung: Zugriffskontrolllisten und Capability-Listen haben in gewisser Weise komplementäre Eigenschaften

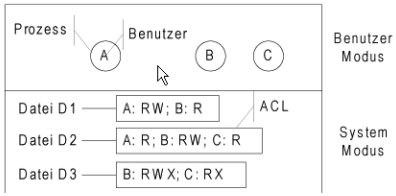
ACLs erlauben das selektive Zurücknehmen von Rechten.

Capabilities können weitergegeben werden.

Zugriffskontrolllisten ("Access Control List", ACL) realisieren die ***spaltenweise*** Speicherung der Schutzmatrix.

jedes Objekt o besitzt seine Zugriffskontrollliste.

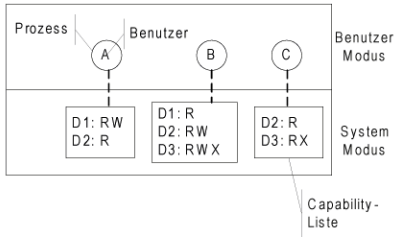
Element einer Zugriffskontrollliste (ACL-Element) besteht aus Paar (Prozess, Zugriffsarten).



Capability-Listen ("Zugriffsausweislisten") realisieren die ***zeilenweise*** Speicherung der Schutzmatrix.

jeder Prozess besitzt eine Menge von Capabilities, die die erlaubten Zugriffe auf Objekte repräsentieren.

Element einer Capability-Liste besteht aus Paar (Objekt, Zugriffsarten).



Capabilities müssen geschützt werden, um Modifikationen durch den Prozess selbst zu verhindern. Alternativen sind

Speicherung im geschützten Bereich des Betriebssystems.

Capabilities sind zwar im Benutzermodus dem Prozess zugeordnet; sie sind jedoch verschlüsselt.

Capabilities können zeitlich begrenzt werden.

Das Konzept der Schutzmatrix wurde von B. Lampson eingeführt. Es verknüpft Schutzdomänen mit den zu schützenden Objekten.

Schutzdomänen

Schutzmonitor

Schutzmatrix ist typischerweise sehr groß und dünn besetzt ⇒ eine direkte Implementierung ist deshalb nicht sinnvoll.

Zugriffskontrollliste

Capability-Liste

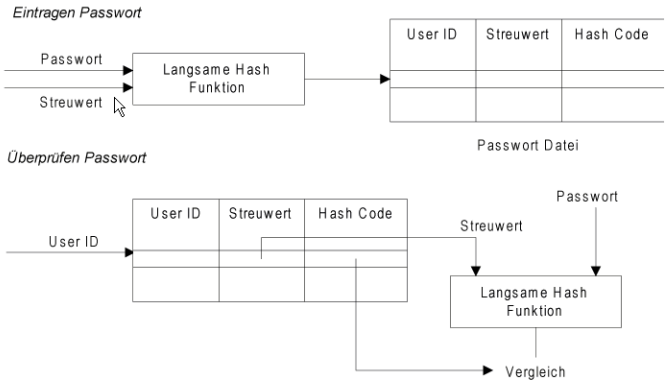
Zusammenfassung: Zugriffskontrolllisten und Capability-Listen haben in gewisser Weise komplementäre Eigenschaften

ACLs erlauben das selektive Zurücknehmen von Rechten.

Capabilities können weitergegeben werden.

Authentifizierung eines Nutzers erfolgt meist über Login-Name und dem zugehörigen Passwort.

Generierung eines Hash Code aus dem Passwort und einem Streuwert fester Länge.



Ziele des Streuwerts

Duplikate von Passwörter sollen in der Passwort Datei nicht erkennbar sein.

Erhöht den Aufwand für offline Attacken auf die Passwort Datei.

nicht erkennbar, ob eine Person dasselbe Passwort auf 2 oder mehreren Systemen nutzt.

Schutz von gespeicherter Information vor Diebstahl, unerwünschter Manipulation und Verletzung der Vertraulichkeit ist ein zentrales Anliegen in allen Mehrbenutzersystemen.

Anforderungen

Ebenen des Zugriffschutzes

Schutzmatrix

Authentifizierung

Ausführung des Applets wird auf einen bestimmten virtuellen Adressbereich beschränkt.

Für eine Sandbox sind die high-order Bits aller Adressen gleich, d.h.

angenommen für einen 32 Bit Adressraum werden 256 Sandboxes auf 16 MByte Grenzen eingerichtet

⇒ für alle Adressen innerhalb einer Sandbox sind die oberen 8 Bits identisch.

jedes Applet erhält zwei Sandboxes: eine Code-Sandbox, eine Daten-Sandbox.

nach dem Laden wird Applet-Code überprüft, ob er Befehle enthält, die ein Verlassen der Sandbox verursachen.

ein Applet, das die Sandbox-Grenzen verletzt, wird zurückgewiesen.

Mobiler Code

Das Internet führt zunehmend zu einer Verbreitung von mobilem Code. Beispiele sind

- Web Seiten mit Applets
- Postscript Dateien
- mobile Software-Agenten (z.B. in Ecommerce Anwendungen).

Ausführung von heruntergeladenem Code birgt Risiken in sich. Methoden (anhand von Applets), um mit dieser Problematik umzugehen:

Sandboxing

Interpretation

Signed Code

Es werden nur Applets von ***vertrauenswürdigen Quellen*** geladen und ausgeführt. Der Applet-Code wird mit einer ***digitalen Unterschrift*** versehen, um zu garantieren, dass der Code der vertrauenswürdigen Quelle nicht verändert wurde.

- digitale Unterschrift basiert auf public-key Verfahren.
- Erzeugung der Unterschrift durch vertrauenswürdige Quelle
  - Hashfunktion erzeugt von Applet-Code eine 128/160 bit Zahl.
  - erzeugte Hashzahl wird mit privatem Schlüssel der Quelle verschlüsselt.
  - digitale Unterschrift wird mit Applet-Code verschickt.
- Überprüfung der Unterschrift
  - Browser führt auf Applet-Code Hashfunktion aus und berechnet selbst Hashzahl.
  - Browser entschlüsselt Unterschrift mit öffentlichem Schlüssel der vertauenswürdige Quelle.
  - berechnete Hashzahl und Hashzahl in Unterschrift müssen übereinstimmen.

Mobiler Code

Das Internet führt zunehmend zu einer Verbreitung von mobilem Code. Beispiele sind

- Web Seiten mit Applets
- Postscript Dateien
- mobile Software-Agenten (z.B. in Ecommerce Anwendungen).

Ausführung von heruntergeladenem Code birgt Risiken in sich. Methoden (anhand von Applets), um mit dieser Problematik umzugehen:

Sandboxing

Interpretation

Signed Code

Generated by Targeteam

Der Entwurf eines BS erfordert ein ingenieurmäßiges Vorgehen. Es gibt jedoch keine wohl-definierten Vorgehensweisen, sondern nur Erfahrungen von Entwicklern bzw. Nutzern.

Einführung

Die Ziele von Betriebssystemen können sich zwischen verschiedenen Systemen unterscheiden, für Server-Systeme, für Laptops, für Smartphones oder für eingebettete Systeme.

Hauptaspekte

Probleme

Schnittstellenentwurf

Ein BS bietet eine Reihe von Diensten, die von Benutzerprozessen in Anspruch genommen werden können.

- Bereitstellen der Dienste über Schnittstellen.

Leitlinien für den Entwurf

Paradigmen der Systemaufrufschnittstelle

Für die Einbindung der Systemaufrufe in Nutzerprogramme kann man zwischen den Ausführungs- und den Datenparadigmen unterscheiden.

Algorithmischer Ansatz

Ereignis-basierter Ansatz

Datenparadigma

Systemaufrufschnittstelle

Weitere Implementierungsaspekte

Trends beim Entwurf von Betriebssystemen

Man kann die folgenden 4 Hauptaspekte unterscheiden

Definierte Abstraktion:

z.B. Thread, Prozess, Datei

Bereitstellen einfacher Operationen:

Operationen zum Manipulieren der zur Abstraktion gehörigen Datenstrukturen.

Ansprechbar über Systemaufrufe.

Sicherstellen der Abgrenzung:

Eingrenzung von Fehlern durch Abgrenzung von Prozessen.

Verwalten der Hardware.

Hardware hat sich gemäß des Moore'schen Gesetzes kontinuierlich verbessert; Betriebssysteme haben zwar mehr Funktionalität, jedoch bzgl. Verfügbarkeit sind sie teilweise schlechter als die alten Systeme.

Betriebssysteme sind sehr komplex.

Betriebssysteme müssen mit Parallelität umgehen.

Betriebssysteme müssen mit feindlichen Nutzern umgehen.

Nutzer möchten Daten mit anderen Nutzern gemeinsam nutzen.

Betriebssysteme existieren eine lange Zeit.

Entwickler wissen oft nicht vorab, wie ihre Systeme genutzt werden.

Betriebssysteme sind portabel für unterschiedliche Hardware entworfen.

Kompatibilität mit älteren Betriebssystem Versionen.

Entwurf von Betriebssystemen

Der Entwurf eines BS erfordert ein ingenieurmäßiges Vorgehen. Es gibt jedoch keine wohl-definierten Vorgehensweisen, sondern nur Erfahrungen von Entwicklern bzw. Nutzern.

Einführung

Die Ziele von Betriebssystemen können sich zwischen verschiedenen Systemen unterscheiden, für Server-Systeme, für Laptops, für Smartphones oder für eingebettete Systeme.

Hauptaspekte

Probleme

Schnittstellenentwurf

Ein BS bietet eine Reihe von Diensten, die von Benutzerprozessen in Anspruch genommen werden können.

Bereitstellen der Dienste über Schnittstellen.

Leitlinien für den Entwurf

Paradigmen der Systemaufrufschnittstelle

Für die Einbindung der Systemaufrufe in Nutzerprogramme kann man zwischen den Ausführungs- und den Datenparadigmen unterscheiden.

Algorithmischer Ansatz

Ereignis-basierter Ansatz

Datenparadigma

Systemaufrufschnittstelle

Weitere Implementierungsaspekte

Trends beim Entwurf von Betriebssystemen

Es existieren eine Reihe von Prinzipien und Empfehlungen

Einfachheit.

Vollständigkeit.

Bereitstellen, was man benötigt und nicht mehr.

Minimum an Mechanismen.

Effizienz.

Nutzergruppen von BS

Endbenutzer

Nutzung von Anwendungen ⇒ grafische Oberfläche des BS

Programmierer

Nutzung der Systemaufrufschnittstelle des BS

Systemadmin

Aufruf von Systemdiensten

Entwurf von Betriebssystemen

Der Entwurf eines BS erfordert ein ingenieurmäßiges Vorgehen. Es gibt jedoch keine wohl-definierten Vorgehensweisen, sondern nur Erfahrungen von Entwicklern bzw. Nutzern.

Einführung

Die Ziele von Betriebssystemen können sich zwischen verschiedenen Systemen unterscheiden, für Server-Systeme, für Laptops, für Smartphones oder für eingebettete Systeme.

Hauptaspekte

Probleme

Schnittstellenentwurf

Ein BS bietet eine Reihe von Diensten, die von Benutzerprozessen in Anspruch genommen werden können.

Bereitstellen der Dienste über Schnittstellen.

Leitlinien für den Entwurf

Paradigmen der Systemaufrufschnittstelle

Für die Einbindung der Systemaufrufe in Nutzerprogramme kann man zwischen den Ausführungs- und den Datenparadigmen unterscheiden.

Algorithmischer Ansatz

Ereignis-basierter Ansatz

Datenparadigma

Systemaufrufschnittstelle

Weitere Implementierungsaspekte

Trends beim Entwurf von Betriebssystemen

Algorithmischer Ansatz

basiert auf der Idee, dass ein Programm eine bestimmte Funktion erfüllen soll, die es im Voraus oder durch Parameter kennt.

Basislogik ist im Code festgelegt

Gelegentliche Systemaufrufe, um

Benutzereingaben zu erhalten

BS-Dienste in Anspruch zu nehmen

Beispiel

```
main () {
    int ...;
    init();
    do_something();
    read (...); /* Systemaufruf */
    do_something_else();
    write (...); /* Systemaufruf */
    continue();
    exit(0);
}
```


Ereignis-basierter Ansatz

nach einer Initialisierung warten Programme dieses Ansatzes auf Ereignisse des Betriebssystems

Anwendungen reagieren auf Ereignisse

Ereignisse werden durch BS erfasst und der Anwendung zugestellt

Ereignisse sind z.B.

Tastendruck

Mausbewegung

Beispiel

```
main () {
    mess_t msg;
    init();
    while (get_message(&msg)) {
        case 1: ...;
        case 2: ...;
        case 3: ...;
        ......
    }
}
```


Wie werden Systemstrukturen und Geräte im Betriebssystem gegenüber dem Programmierer präsentiert?

Beispiel Unix

"Alles" ist eine Datei

Vereinheitlichung von Dateien, E/A-Geräte und Pipes

Zugriff über Dateioperationen

fd1 = open("file1", O_RDWR)

fd2 = open("/dev/tty", O_RDWR)

Beispiel Windows 2000

"Alles" ist ein Objekt.

Entscheidung bzgl. der Bereitstellung unterschiedlicher Varianten von Systemaufrufen

Z.B. Lese-Systemaufruf: read_file, read_process, read_tty, ...

Besser: nur Bereitstellung des allgemeinen Falls

Z.B. Lese-Systemaufruf: read

Restliche Varianten über Bibliotheksfunktionen auf den allgemeinen Systemaufruf abbilden

Sichtbarkeit von Systemaufrufen

Trennung von Systemaufrufen und Bibliotheksaufrufen

⇒ Ermöglicht übersichtliche Darstellung der vorhandenen Systemaufrufe