7. Sicherheit

- Überblick
  - 7.1 Grundlagen der Sicherheit
  - 7.2 Benutzer-Authentifizierung
  - 7.3 Angriffe von innerhalb des Systems
  - 7.4 Angriffe von außerhalb des Systems
  - 7.5 Schutzmechanismen

7.1 Grundlagen der Sicherheit

- Aus Sicherheitsperspektive müssen von einem Betriebssystem folgende Eigenschaften gewährleistet werden
  - Datenvertraulichkeit: Geheime Daten müssen geheim bleiben
  - Datenintegrität: Unautorisierte Benutzer dürfen nicht in der Lage sein, Daten zu manipulieren (Ändern, Löschen, Austauschen, …)
  - Systemverfügbarkeit: Sicherstellung einer Mindestverfügbarkeit unabhängig von der aktuellen Lastsituation
  - Datenschutz: Schutz vor Missbrauch der personenspezifischen Daten
- Korrespondierende Bedrohungen
  - Aufdeckung, Manipulation, Denial-of-Service-Angriffe, Speicherung und Kombination persönlicher Daten
- Außerdem Datenverlust durch Betriebsstörungen (*Höhere Gewalt wie Feuer*), HW- und SW-Fehler, Fehlbedienung, …
  - Aufbau von Redundanz, Backupstrategien

Besonders einfach sind Mailserver anzugreifen

Eindringlinge

- Unterschiedliche Kategorien von Eindringlingen, die ein oder mehrere Schutzziele gezielt angreifen
  - Herumschnuffeln: Lesen von Daten anderer Benutzer, aus reiner Neugier oder um persönlichen Vorteil zu ziehen
    - Nicht technisch-versierte Benutzer: Ausnutzen bekannter Lücken, Missbrauch der eigenen Rechte
    - Technisch-versierte Benutzer: gezielte Angriffe mit großem Know-how und Zeitaufwand
  - Erzielen eines wirtschaftlichen Nutzens wie Einbruch bei Bankcomputern und Geldüberweisung, …
  - Militärische Spionage oder Wirtschaftsspionage
    - Einbruch in Firmenrechner, um Programme, Geschäftsgeheimnisse, patentwürdige Ideen, Technologien, … zu stehlen

      $\Rightarrow$ Einsatz mit vielen Mitteln und ernstzunehmenden Spezialisten
  - Viren: In der Regel wird wahllos Schaden angerichtet

Wie dringen Eindringlinge ein:
- niedrigste Kategorie: bekannte Lücken ausnutzen
- mittel: Emailanhänge etc -> unvorsichtige Nutzer ausnutzen; unbekannte Lücken ausnutzen
- hoch: Software mit Lücken erstellen & einschleusen (z.B. Firmware)

Kryptografie

- Ziel: Eine Nachricht in Klartext wird verschlüsselt und so in einen Chiffretext überführt, dass nur autorisierte Personen eine Rückgewinnung des Klartextes durchführen können
  - Verschlüsselungsfunktion/Schlüssel zur Verschlüsselung (Encryption Key K_E)
  - Entschlüsselungsfunktion/Schlüssel zur Entschlüsselung (Decryption Key K_D)
- Symmetrische Kryptografie (*Secret-Key-Kryptografie*) z.B. Verschlüsselung von Handy-Telefonaten
  - Schlüssel zur Verschlüsselung lässt sich aus dem Schlüssel zur Entschlüsselung bestimmen und umgekehrt Problem: Pro Kommunikation ein Schlüssel, dieser muss ausgetauscht werden
- Public-Key-Kryptografie
  - Benutzer lässt ein Schlüsselpaar (Private Key K_D, Public Key K_E) so erstellen, dass $K_D K_E(m) = K_E K_D(m) = m$
    - Public Key wird veröffentlicht (Webseite, Keyserver, Mail, ...) $\Rightarrow$ Jeder mögliche Sender kann diesen Schlüssel bekommen
    - Private Key bleibt beim Benutzer und ist geheim

Schwachstelle: Person kann sich als andere Person ausgeben
-> Lösung: (sende von A zu B) Nachricht wird mit Signatur gesendet, die mit PublicKey von B und PrivateKey von A verschlüsselt wurde
-> B entschlüsselt und vergleicht unterschrift

Einwegfunktionen und digitale Signaturen

= funktioniert nur in eine Richtung

Wegen vielen Operationen, kann die Funktion nicht rekonstruiert werden -> keine Logik, sondern wilde Operationen auf der Eingabe (Bits umkehren etc)

- Einwegfunktionen: Wird x mit der Einwegfunktion f verschlüsselt, so ist die Bestimmung von x aus dem Ergebnis f(x) praktisch unmöglich
  - Verschlüsselung von Passwörtern
  - Challenge/Response auf Chip Karten $\Rightarrow$ Berechungsvorschrift f geheim
- Digitale Signaturen: Eindeutige Senderidentifikation

einfacher als komplette Verschlüsselung

  - Dokumentbearbeitung mit Einweg-Hash-Funktion $\Rightarrow$ Ausgabe fester Länge
    - MD5 (*Message Digest*): 16 Byte langes Ergebnis
    - SHA (*Secure Hash Algorithm*): 20 Byte langes Ergebnis
  - Signierung des Hash-Wertes und evtl. Verschlüsselung der gesamten Nachricht
  - Bei Ankunft
    - Der Text wird entschlüsselt und der Hash-Wert wird berechnet
    - Signatur wird entschlüsselt und die beiden Hash-Werte verglichen
    - Stimmen die Werte nicht überein, so wurde der Text, die Signatur oder beides manipuliert

Einweg-Hash: komprimiert Eingabetext bis er ins Ausgabeformat passt
-> kombinieren mit Verschlüsselung mit Public Key (also den Hash-Wert verschlüsseln); Empfänger wendet auf Nachricht auch Hash an und verschlüsselt ebenfalls mit seinem Key -> stimmen beide überein? -> dann alles gut, sonst Nachricht manipuliert

7.2 Benutzer-Authentifizierung

- Feststellung der Benutzeridentität, wenn sich dieser in den Computer einloggen will
- Realisierung der Authentifikation durch etwas, was der Benutzer
  - Weiß ⇒ z.B. Passwortabfrage
  - Besitzt ⇒ z.B. Chipkarten z.B. auch mobile TAN
  - Ist ⇒ z.B. biometrische Abfragen
- Authentifizierung durch Passwörter
  - Einfachste Implementierung basiert auf einer zentralen Liste, die den Loginnamen und das Passwort enthält
  - Vergleich der eingegebenen und gespeicherten Daten: Bei Übereinstimmung wird der Zugriff gestattet

ca 80% der Passwörter sind durch Dictionary-Attacken zu knacken

Schwachstellen bei passwortgeschützten Systemen

- Angriffe durch Ausprobieren von Kombinationen (Login, Passwort)
  - Hilfe durch Wörterbücher, Namenslisten, …
  - Untersuchungen: über 80% aller Passwörter sind unsicher
- Vorgehensweise
  - Ermittlung aktiver, vernetzter Rechner, z.B. durch ping w.x.y.z ⇒ aktive Rechner antworten und dienen als Ziele
  - Ausprobieren der Passwortkombinationen wie z.B. ermöglicht Telnet mehrfache Eingabe von Login/Passwort
  - Oft Begrenzung möglicher sukzessiver Versuche ⇒ Starten vieler paralleler Threads und Verteilung auf mehrere Rechner
  - Sicherheitslücke nicht vollständig zu stopfen ⇒ Telnet ist mittlerweile oft abgeschaltet oder nur von eingetragenen IP-Adressen erlaubt
  - Installation von Paket-Sniffern zur Überwachung des Netzwerkverkehrs und Aufspüren von interessanten Mustern

Passwortsicherheit in UNIX

- Passwort als Schlüssel zur Verschlüsselung eines festen Datenblocks
- Passwortdatei: je eine Zeile mit Attributen und verschlüsseltem Passwort pro Benutzer
- Beim Einloggen: Verschlüsselung sofort nach Passworteingabe
  - ⇒ Vergleich der verschlüsselten Sequenzen bei Authentifizierung
  - ⇒ Niemand (auch nicht Sysadmin) kann die Passwörter in Klartext sehen
- Allerdings ist der Verschlüsselungsalgorithmus bekannt
  - ⇒ Wörterbuch kann entsprechend verschlüsselt und mit den Einträgen der Passwortdatei verglichen werden
- Probleme früherer Versionen
  - ➢ Passwortdatei (/etc/passwd) mit verschlüsselten Passwörtern auf der Festplatte für Benutzer sichtbar
  - ➢ Datei wird kopiert und zum Offlineausprobieren der Passwörter genutzt
  - ➢ Mittlerweile getrennte Speicherung der Passwörter in Shadow-Datei

/etc/passwd **und** /etc/shadow

```
# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
rene:x:500:100:Rene:/home/rene:/bin/bash
christian:x:501:100:Christian:/home/christian:/bin/bash
```

user:password (x =>/etc/shadow) user-id:group-id:complete name:home directory: default shell

```
# ls -la /etc
-rw-r--r--    1 root     root          2687 Apr  7  2001 passwd
-rw-r-----    1 root     shadow        1291 Mär 22  2002 shadow

# cat /etc/shadow
root:fntzjTtoSqExc:11749:0:10000::::
rene:YJMasdgfEN0M:11749:0:99999:7:::
christian:qdoncRTvonfDTVgh:11750:0:10000::::
```

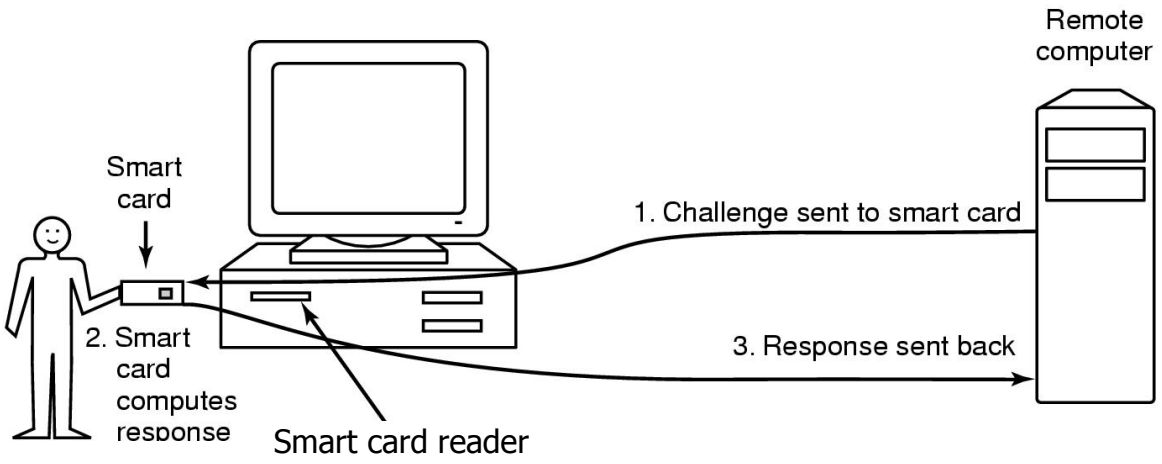
username:coded pass: last_change:min_days: max_days:warn_days: inactive_days: expire_date

Verbesserung der Passwortsicherheit

- Verzögerungsstrategien
  - Erweiterung der Passwörter um eine n-Bit lange (UNIX n=12) Zufallszahl vor der Verschlüsselung $\Rightarrow$ Datei mit zu testenden Passwörtern wird um Faktor 2^n größer (Salt-Technik)
  - Verschlüsselung der kompletten Passwortdatei. Eine spezielle Funktion liefert Einträge auf Anfrage $\Rightarrow$ verlangsamter Zugriff
- Schwer zu ermittelnder Passwörter, etwa mindestens 7 Zeichen, nicht im Wörterbuch, Klein/Großschreibung, …
  - Erzwingen einer regelmäßigen Änderung des Passworts
- Einmal-Passwörter: Passwortliste zur sequentiellen Bearbeitung
- Challenge-Response-Verfahren
  - Abfrage von vorher eingegebenen und verschlüsselt gespeicherten Fragen, z.B. Name der Klassenlehrerin in der achten Klasse?
  - Vereinbarter Algorithmus muss mit einem vom Server gesendeten Wert ausgeführt werden $\Rightarrow$ Ergebnis = Passwort

Authentifizierung durch Gegenstände

- Benutzer muss etwas haben: Schlüssel ⇒ Türschlösser
- Benutzer muss etwas haben und wissen: Karte + PIN
  - Magnetstreifenkarten: 140 Byte, oft ist das verschlüsselte Passwort dabei ⇒ riskant, da Lese/Schreibgeräte weit verbreitet
  - SmartCards: kleine CPU vorhanden
    - Aufbau einer Verbindung zum Server, der dann Daten sendet
    - SmartCard verarbeitet die Daten und sendet das Ergebnis als Passwort zurück ⇒ Berechnungsvorschrift geheim



Biometrische Authentifizierung

- Messung (*Registrierung*) und Vergleich (*Identifizierung*) von charakteristischen, schwer zu fälschenden Merkmalen des Benutzers
- Speicherung der Merkmale auf zentralem Rechner/SmartCard
- Entscheidend: Güte der Merkmale
  - Weite Streuung (Haarfarbe kein gutes Merkmal)
  - Keine gravierende Änderung im Laufe der Zeit
  - Erkennung der Veränderungen (Sonnenbräune, Makeup, Erkältung, ...)
- Typische Merkmale
  - Fingerlänge, Fingerabdrücke $\Rightarrow$ Probleme bei Gipsvorlagen
  - Scann der Netzhaut $\Rightarrow$ Probleme beim Abfotografieren
  - Automatische Unterschriftenanalyse $\Rightarrow$ Probleme bekannt
  - Stimmbiometrie $\Rightarrow$ Täuschung durch Stimmaufnahmen
- Kombination verschiedener Merkmale erhöht die Sicherheit, reduziert aber die Akzeptanz der Sicherheitsmaßnahmen unter den Benutzern

7.3 Angriffe von innerhalb des Systems

- Passwort geknackt ⇒ unterschiedliche Schäden möglich
  - Eher sichere Systeme: lediglich der Benutzer, dessen Passwort geknackt wurde, erleidet Schaden
  - Eher unsichere Systeme: der erste Zugang dient lediglich als Startpunkt, um wesentlich größeren Schaden einzurichten
- Beispiel Trojanische Pferde
  - Scheinbar harmloses Programm, dessen Code aber unerwartete und unerwünschte Funktionalität ausführt
  - Vorgehensweise
    - Tarnen des Programms (Spiele, MP3-Player, …) ⇒ Benutzer laden und installieren das Programm freiwillig
    - Wenn das Programm einmal läuft, kann es alles tun, wozu der aktuelle Benutzer berechtigt ist, z.B. Daten übers Netz versenden, Dateien modifizieren, teure Nummern anrufen, …

      Keylogger, Screenshots

= Phishing

Login-Spoofing

- Dient dem Ausspionieren von Passwörtern bei der Einloggprozedur
  - Programm bildet die Eingabemaske exakt nach
  - Falls der Benutzer die Fälschung nicht mitbekommt, so gibt er sein Login/Passwort wie gewohnt ein
  - Daten werden in Datei geschrieben und die Shell beendet sich ⇒ Erst jetzt wird der richtige Einloggbildschirm dargestellt
  - Benutzer denkt, er hätte sich vertippt und wiederholt den Einloggprozess ⇒ Fälschung bleibt unbemerkt
  - Die gespeicherten Daten können später in Ruhe abgerufen werden
- Schutzmöglichkeit
  - Starten der Einloggprozedur mit einer Tastenkombination, die von einem Benutzerprogramm nicht abgefangen werden kann
  - Beispiel Windows
    - CTRL+ALT+DEL loggt aktuellen Benutzer aus, startet Loginprozedur
    - Umgehung dieses Mechanismus ist nicht möglich

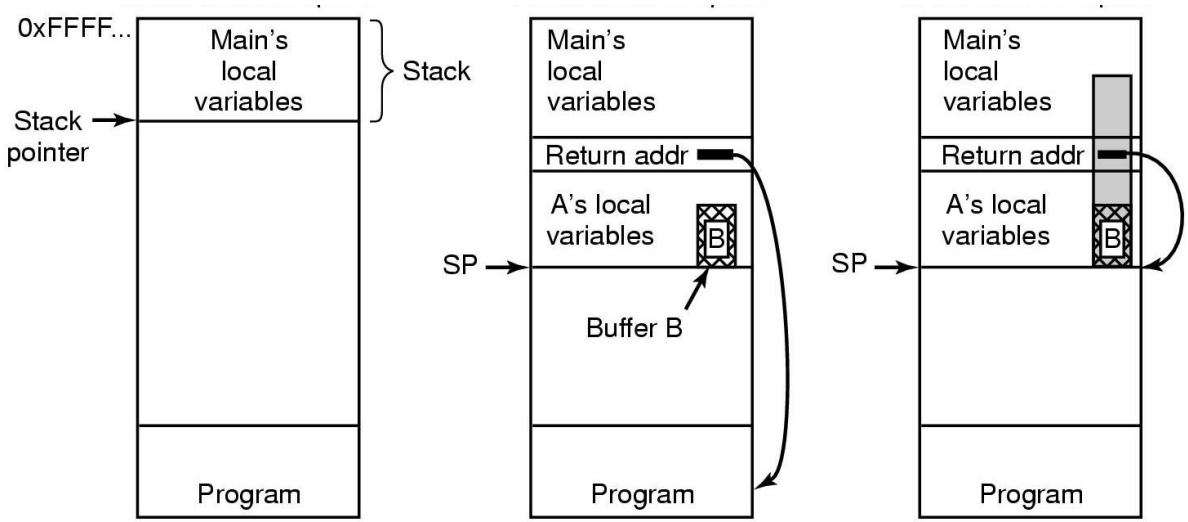
Logische Bomben und versteckte Hintertüren

- Logische Bomben = vom Programmierer beim Erstellen des Programms eingebrachter Code, der Schaden einrichten kann
  - Oft zur Erpressung des Arbeitsgebers eingesetzt
  - Solange das Passwort des Entwicklers täglich kommt oder sein Name auf der Gehaltsliste steht, passiert nichts
  - Bei Entlassung: Code wird aktiviert und richtet Schaden an
- Versteckte Hintertüren
  - Veränderung der Einloggprozedur, so dass mit einem bestimmten Loginnamen das Einloggen jederzeit und auf allen vom Hersteller ausgelieferten Systemen möglich ist ⇒ Hintertür umgeht den gesamten Authentifizierungsprozess
- Vermeidung von logischen Bomben/versteckten Hintertüren durch Einführung eines konsequenten, ausgiebigen Code Reviews möglich
  - Entwickler erklärt den Code Zeile für Zeile den restlichen Teammitgliedern
  - Ein oder mehrere Entwickler überprüfen den Code der anderen Teammitglieder ⇒ siehe Software Engineering

oder OpenSource Software verwenden

Pufferüberläufe (*Buffer Overflows*)

- Umlenkung der Rücksprungadresse auf ein eingeschleustes Programm
  - Bei Funktionsaufrufen: Rücksprungadresse und Parameter auf den Stapel
  - Für bestimmte Parameter – z.B. einen Dateinamen – wird Speicherplatz gemäß der Definition des Strings reserviert
  - Wird diese Grenze durch einen extralangen Namen überschritten, so wird die Rücksprungadresse überschrieben und somit abgeändert
    - Sprung in fremden/gesperrten Adressraum ⇒ Absturz
    - Pufferinhalt = Programm, das ausgeführt wird ⇒ Großer Schaden



Pufferüberläufe (2)

- Wo kann dieser Fehler auftreten?
  - Überall, wo Felder fester Größe für Benutzereingaben benutzt werden (Dateinamen, Umgebungsvariablen, Datenabfragen, …)
- Wie findet man solche Sicherheitslücken?
  - Extremeingaben bei allen Aufforderungen: Stürzt das Programm ab, so wahrscheinlich wegen eines Pufferüberlaufs
  - Analyse des Fehlerreports (z.B. core Datei) kann u.U. Rückschlüsse auf den Stackaufbau zu dem Zeitpunkt erlauben
  - Noch einfacher geht es bei offenen Quellen (in beiden Richtungen, sowohl Ausnutzen als auch Verhindern)
- Verhinderung: Nutzung von Routinen, die Stringlänge limitieren
  - fgets anstatt von gets
  - strncpy anstatt von strcpy

7.4 Angriffe von außerhalb des Systems

- Vernetzung ⇒ vielfältige Angriffsmöglichkeiten von außerhalb
- Häufige Angriffe
  - Denial-of-Service-Angriffe: Rechnerüberflutung mit sinnlosen Anfragen
  - Viren: Programme, die sich durch Anhängen an andere Programme / Dokumente replizieren können und evtl. Schaden anrichten
    - Companion Viren: Ähneln – bis auf die Endung – einem existierenden Programm und werden zuerst ausgeführt
    - Überschreibende Viren: Ersetzen existierende Programme
    - Parasitäre Viren: Hängen sich an Programm an, werden ausgeführt und erlauben anschließend die Ausführung des tatsächlichen Programms
    - Makroviren: Versteck in Anwendungen, die benutzerdefinierte Makros aufnehmen und ausführen können
    - Speicherresidente Viren, Bootsektorviren, Treiberviren, Quellcodeviren, …
  - Phishing, …

Netzwerksicherheit ist ein sehr umfangreiches Thema ⇒ Siehe Vorlesungen im Bachelor / Masterbereich, aktuell fehlen Grundlagen der Netzwerke

7.5 Schutzmechanismen

- Grundlage für die Sicherheit ist das Domänenkonzept (*domain*)
  - Domäne = Menge von Paaren (Objekt, Rechte)
  - Jedes Paar spezifiziert ein eindeutig identifiziertes Objekt (HW, Prozesse, Dateien, Semaphore, …) und die darauf ausführbaren Operationen
  - Recht = Erlaubnis zur Ausführung einer Operation
  - Domäne kann einem Benutzer entsprechen, aber auch allgemeiner Natur sein
  - Ein Prozess läuft zu jedem Zeitpunkt in einer einzigen Schutzdomäne
  - Domänenwechsel ist möglich, die dazugehörigen Regel sind hochgradig vom aktuellen System abhängig
- Domänenkonzept bei UNIX
  - Domäne eines Prozesses wird durch BenutzerID (UID) und GruppenID (GID) festgelegt ⇒ für jede Kombination (UID, GID) lässt sich eine Liste mit allen benutzbaren Objekten (HW, SW, …) erstellen
  - Prozessaufteilung in Benutzer- und Kernteil, die in unterschiedlichen Domänen laufen ⇒ Systemaufruf verursacht einen Domänenwechsel

Realisierung des Domänenkonzepts

- Konzeptionell: Modellierung der Übersicht über Domänen und zulässige Operationen für verschiedene Ressourcen durch eine Matrix
  - Ein Objekt pro Spalte, eine Domäne pro Zeile
  - Zellen enthalten – falls vorhanden – die zulässigen Operationen
  - Domänen sind selbst Objekte ⇒ Modellierung des Übergangs von einer Domäne in eine andere Domäne
- Matrixdarstellung ineffizient, da zu viele leere Felder vorhanden ⇒ Speicherung nur der Spalten (*ACL*) oder der Zeilen (*Capabilities*)

Pro Benutzer eine Liste mit Dingen, die er darf (für Nutzer 3 z.B. nur noch 3 Einträge) -> in den meisten OS so
Pro Datei eine Liste mit Zugriffsrechten der Benutzer (vom Nutzer nicht zu ändern!)

Object

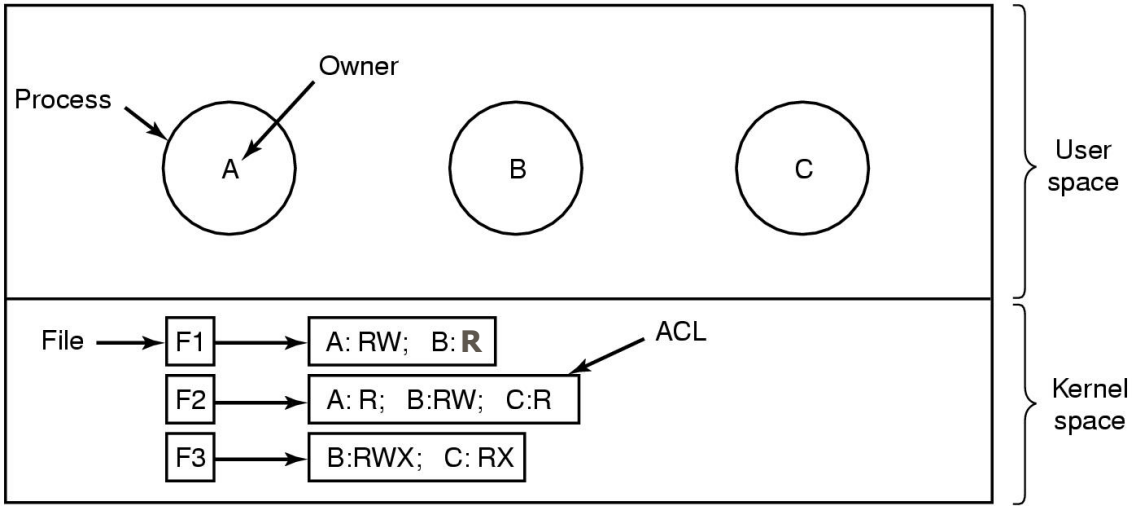
Domain	File1	File2	File3	File4	File5	File6	Printer1	Plotter2	Domain1	Domain2	Domain3
1	Read	Read Write								Enter	
2			Read	Read Write Execute	Read Write		Write				
3						Read Write Execute	Write	Write			

Realisierung des Domänenkonzepts (2)

- Zugriffskontrolllisten (*Access Control Lists*, ACL)
  - Eine Liste pro Objekt spezifiziert, welcher Prozess, Benutzer, … (allgemein: Subjekt) welche Operation (für dieses Objekt) ausführen darf
    - Unix: Unterscheidung zwischen Rechten für den Eigentümer der Datei, die Mitglieder einer Gruppe und die restlichen Benutzer
    - Windows: Beliebig viele Einträge ⇒ genauere, fein spezifizierte Kontrolle
- Capabilities: Spezifizieren die Berechtigung eines Subjekts, auf ein bestimmtes Objekt zuzugreifen

  ⇒ jedem Subjekt wird seine eigene Domäne angehängt
  - Verwaltung in einer Liste, wobei jeder Eintrag folgendes angibt
    - Referenz auf das Objekt
    - Menge der zugelassenen Operationen
  - Liste wird z.B. durch Verschlüsselung geschützt

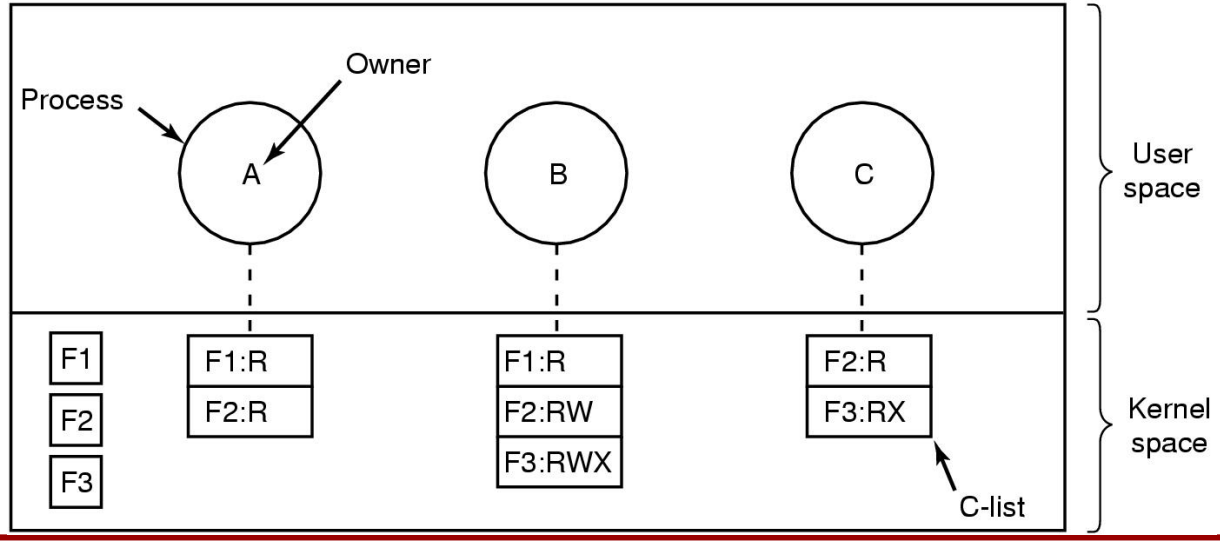
Zugriffskontrolllisten



- Prozesse in unterschiedlichen Domänen (A, B, C) und Dateien F1, F2, F3 mit eigenen ACL-Listen, welche die Zugriffsrechte für Subjekte in den Domänen spezifizieren (R=Read, W=Write, X = eXecute)
- Bei UNIX Definition von Benutzern (UID) und Gruppen (GID)
  - Ein Benutzer kann in mehreren Gruppen eingetragen sein, z.B. sysadmin, postmaster, mitarbeiter, …
  - Beim Zugriff ⇒ Überprüfung, ob die Kombination (UID, GID) in der ACL vom Objekt enthalten ist

Capabilities (C-Listen)

- Jedes Subjekt hat Liste mit Rechten für bestimmte Objekte ⇒ Capability = Objektbezeichner + Bitmap für Zugriffsrechte
- Schutz der C-Listen
  - Tagged Architecture: Spezielles Bit zeigt an, ob das Speicherwort eine Capability enthält ⇒ Änderung nur im Kernmodus möglich
  - C-Listen werden innerhalb des BS-Kerns gehalten und über ihre Position referenziert
  - Verschlüsselte C-Listen im Benutzermodus ⇒ Manipulation durch Benutzer nahezu unmöglich



Sicherheitsanforderungen

- Trusted Computer System Evaluation Criteria (TCSEC): Orange Book (1983)
  - ➢ Sicherheit von Rechnersystemen von US Verteidigungsministerium
- Unterschiedliche Sicherheitsklassen definieren Anforderungen an Sicherheitspolitik, Verantwortlichkeit

  D – minimaler Schutz (niedrigste Klasse)

  C1 – Sicherheitsschutz nach Ermessen (Aktuelle Einstufung von Linux)

  C2 – Kontrollierter Zugriffsschutz (Aktuelle Einstufung von Windows)

  B1 – Sicherheitsschutz mit Etiketten

  B2 – Strukturierter Schutz

  B3 – Sicherheitsdomänen

  A1 – verifizierter Entwurf (die höchste Klasse)

Warum Windows sicherer eingestuft als Linux?
Damals erfolgte Anmeldung über CTRL+ALT+DEL
-> Sicherheit gegen Spoofing

C2-Standard Umsetzung

- Sicheres Anmelden mit Antispoofing-Maßnahmen
  - Strg-Alt-Entf wird immer vom Tastaturtreiber erkannt und startet darauf ein Systemprogramm mit originalem Anmeldebildschirm
- Frei einstellbare Zugriffskontrollen
  - Besitzer einer Datei vergibt Rechte, eine Datei kann für verschiedene Benutzergruppen mit verschiedenen Rechten ausgestattet sein
- Privilegierte Zugriffskontrollen
  - Administrator hat Zugriff auf Dateien von einem „normalen User", kann Dateirechte überschreiben (ändern)
- Schutz des Adressraumes für jede einzelne Person
  - Jeder Prozess hat eigenen, abgeschotteten Adressraum im Speicher
- Stackseiten müssen vor Einlagerung mit Nullwerten belegt werden
  - Keine Information über vorherigen Besitzer vom Stack abrufbar
- Security auditing
  - Aufzeichnung (Log) über sicherheitsrelevante Ereignisse

hatte Win früher als Linux

Vielen Dank!

Herzlichen Dank für die Aufmerksamkeit
und viel Erfolg bei der Klausur

See you in

Verteilte Systeme

Cloud Computing

Betrieb komplexer IT-Systeme