Security

Peter Puschner
Institut für Technische Informatik
peter@vmars.tuwien.ac.at

- Definition und Ziele
- Bedrohungen
- Betriebssystemmechanismen

Security

- Strategien, Vorkehrungen und Tools, um die Vertraulichkeit, Integrität und Verfügbarkeit von Information in einer Organisation zu gewährleisten.
  - Relevante Bedrohungsszenarien?
  - Abläufe und Strategien
    (Akteure und Prozesse,
    Wer darf welche Information wie manipulieren?)
  - Computerspezifische Fragen
    (Wie werden Sicherheitsanforderungen realisiert?)
  - Basis: Vertrauen – Vertrauenswürdigkeit?

Security Objectives (CIA-Triad)

- Confidentiality (Secrecy)
  - Geheimhaltung
    (z.B.: Wer darf Daten sehen?)
- Integrity
  - Daten entsprechen dem Sollzustand
    (z.B.: Wer darf Daten verändern?)
- Availability
  - Verfügbarkeit der Daten

Security Concerns

… Authenticity

- Korrektheit der Identität

… Accountability

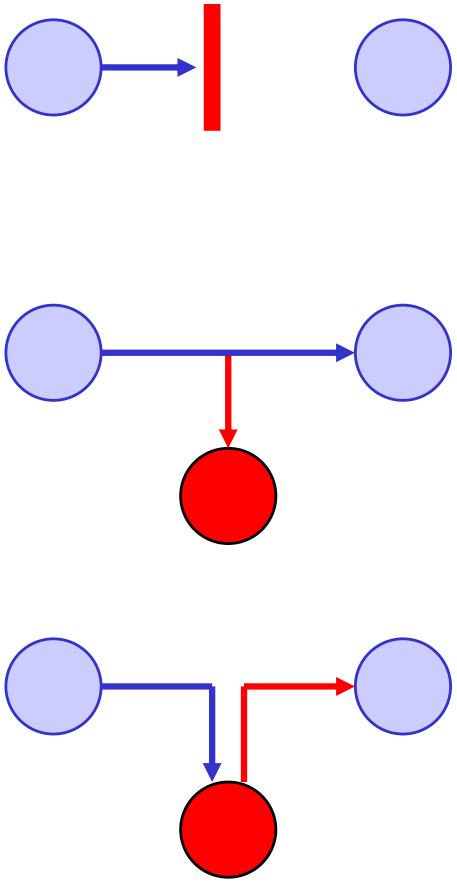
- Nachvollziehbarkeit, Protokollieren von Zugriffen

Types of Security Threats

- Passive Threats
  - Abhören oder Monitoring von Information ohne Wissen des Betroffenen: Confidentiality Violation
- Active Threats
  - Aktive Manipulation von System und Daten
  - Beeinträchtigung von Integrität und Service

Security Threats

- Denial of Service (Interruption)
  - Vorübergehende oder permanente Unterbrechung eines Services (physikal. Zerstörung, Überlast, etc.)
- Exposure / Interception
  - Nicht autorisierter Lesezugriff
- Modification / Fabrication
  - Verletzung der Datenintegrität (Änderung/Generierung von Daten)
  - Theft of Service: unautorisierte Verwendung von Ressourcen



Goals versus Threats

Goal	Threat
Confidentiality	Exposure / Interception
Integrity	Modification / Fabrication
Availability	Denial of Service

Intrusion

- Ziele
  - Verschaffen des Zugangs zu einem System
  - Erhöhung der Privilegien
- Methode
  - Ausnützung einer Sicherheitslücke
  - Aneignung eines Passworts

Intruders (Adversaries)

Verschiedene Kategorien von Szenarien:

- Gelegenheitsattacken von nicht versierten Benutzern
- gelegentliche Attacken von technischen Insidern
- gezielte Versuche der Bereicherung
- Industrie-, Militärspionage, ..., Cyber War

Malware

- **Virus**: In einem Programm versteckter Code, der sich selbst in andere Programme kopiert
  - ev. zusätzlich destruktives Verhalten
  - Denial Of Service (DOS) durch Ressourcenblockierung
  - Distributed Denial Of Service (DDOS) Attacken
  - Key Logger
- **Worm**: Programm, das sich selbst repliziert und Kopien seines Codes über ein Netz an andere Computer sendet (mail, remote login);

Malware

- **Trojan Horse**: Programm mit gewünschter Funktionalität, das versteckten Code mit bösartiger Funktionalität enthält
  - oft vom Benutzer selbst erworben (z.B. Download)
- **Logic Bomb**: Programmstück, das sich selbst bei Auftreten einer Bedingung aktiviert
  - z.B. Zeitablauf: kein Login eines Mitarbeiters für eine gewisse Dauer → Start von bösartigem Code
- **Trapdoor**: Geheimer Einstiegspunkt, Umgehung der Zugriffskontrolle

Malware

- **Port Scan**: automatisierter Versuch des Verbindungaufbaus zu unterschiedlichen Ports, um bekannte Fehler auszunützen
- **Denial of Service**: Überlastung eines Rechners, sodass dieser kein sinnvolles Service mehr zur Verfügung stellen kann
  - **Distributed Denial of Service** (DDoS): gleichzeitige Attacke von unterschiedlichen Rechnern

Typ. Methoden von Attacken

- Anfordern und Auslesen von Speicher (Memory, Disk, …)
- Aufruf unerlaubter System Calls bzw. erlaubter Calls mit unerlaubten / “sinnlosen” Parametern
- Verwendung von Abbruchtasten (DEL, BREAK, etc.) während des Loginprozesses
- Modifikation von BS-Datenstrukturen, die im User Space gehalten werden
- Fälschung der Login-Routine, Abhören unverschlüsselter Verbindung (z.B. telnet)

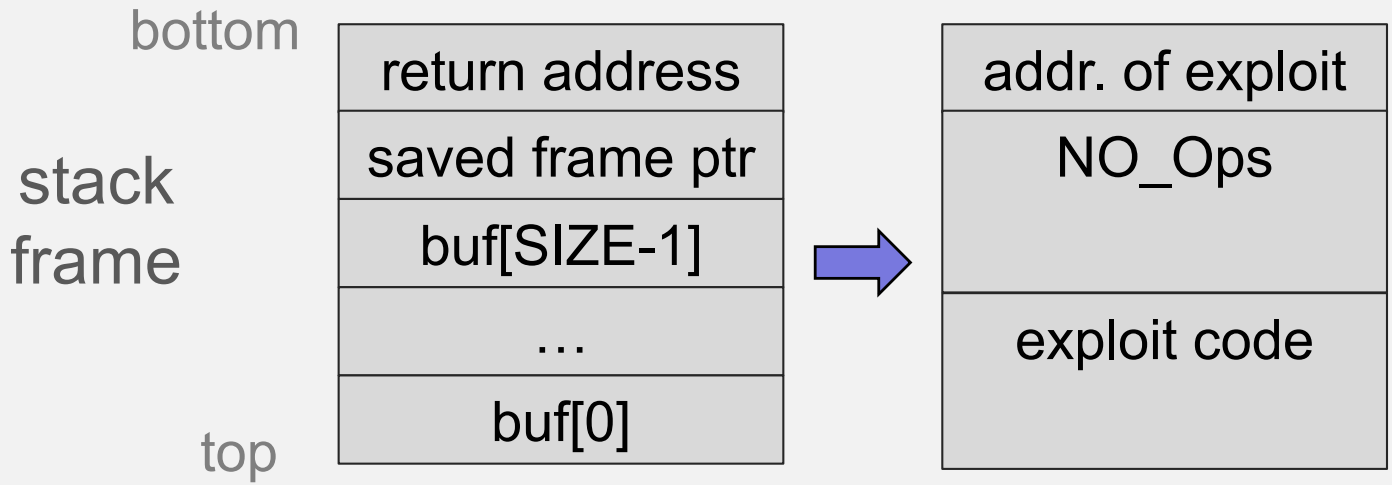
Typ. Methoden von Attacken (2)

- Wenn das Manual sagt: “Do not do X”, möglichst viele Varianten von X ausprobieren
- Bitte an Systemprogrammierer, bestimmte Sicherheitsüberprüfungen für eigenen Account auszuschalten
- Social Engineering (z.B. Phishing, Bestechung)

Bsp. Stack/Buffer Overflow Attack

- Über Pufferende schreiben und damit Stack verändern – wenn Buffer Overflow nicht geprüft wird.
- Exploit Code am Stack positionieren.
- Returnadresse am Stack auf Exploit-Code setzen

Code ohne Size Check: `strcpy(buf, argv[1]);`



Design Principles for Security

- Open Design: nicht “Security by Obscurity”
- Default-Einstellung: keine Berechtigung
- Least Privilege
- Economy of Mechanisms:
  Einfachheit der Sicherheitsmechanismen,
  Implementierung auf möglichst niedriger Ebene
- Acceptability: akzeptierbare Mechanismen
- Überprüfung der gegenwärtigen Berechtigung
- Complete Mediation: Kontrolle aller Zugriffe auf Ressourcen (auch Ausnahmesituationen)

User Authentication

- Sicherstellen, dass der Benutzer (bzw. Benutzerprozess) der ist, der er zu sein vorgibt
  - Besitz eines Schlüssels: physikalischer Schlüssel, Chipkarte, etc.
  - Überprüfung von Attributen: z.B. Fingerabdruck, Iris, Gesichtserkennung – Problem der Akzeptanz
  - Abfrage von Wissen: Passwort
    gängigster Mechanismus
    leicht zu implementieren
    Problem: Qualität von Passworten

Passwörter

- Suchraum
  - Anzahl Passwords mit 7 druckbaren Zeichen:
    $95^7 \approx 7*10^{13}$ (vgl. $26^6 \approx 3*10^8$)
  - Speichern von verschlüsselten Passwörtern,
    1000 Verschlüsselungen pro Sekunde
    ➪ 2000 Jahre für Probieren aller Verschlüsselungen;
    Speicherplatzbedarf für alle Passwörter ... ?
  - Vergrößerung des Suchraums durch *salt*:
    n sichtbare bits, die bei Verschlüsselung mit-
    verwendet werden
    ➪ vergrößert Suchraum um 2^n (Bsp. Unix: 12 bit salt)

Standard-Passwortattacken

- Default Passwort
- Durchprobieren aller kurzen Passwörter
  Bsp.: Buchstabenkombinationen mit 6 Chars
- Verwendung von Wörterbüchern, Namen
- Wörterbuch + typische Ersetzungen (i⇨1, o⇨0), Groß/Kleinschreibung, Umkehrung, etc.
- User-spezifische Information (Geburtsdatum, KFZ-Kennzeichen)

Passwortattacken - Maßnahmen

- One-time Password (TAN)
- Zeitablauf von Passwörtern
- Challenge Response Protokolle

- Zeitverzögerung bei Logins (speziell nach gewisser Anzahl von Fehlversuchen)
- Logging, Anzeige der Daten des letzten Login

Protection

Sicherstellen des kontrollierten Zugangs zu Programmen und Daten auf einem Computersystem.

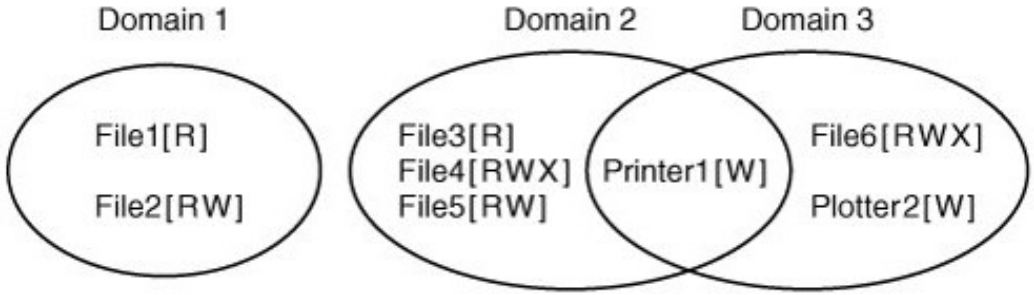
- Einschränkungen
  - Zugriff auf Daten
  - Verwendung der Daten
- Schutz der Integrität der Daten gegen
  - Zerstörung
  - bösartigen Zugriff

Protection Domains

Objekte: CPU, Speichersegmente, Files, …

Zugriffsrecht: Paar (Objekt, Rechte)

Domain: Menge von Zugriffsrechten



Unix: Domain v. Prozess durch (UID, GID) definiert;

Änderung der Domain:

- User Part vs. Kernel Part eines Prozesses
- *exec* von File mit gesetztem SETUID, SETGID Bit

Access Matrix

Domain – Object – Access rights

	Object							
Domain	File1	File2	File3	File4	File5	File6	Printer1	Plotter2
1	Read	Read Write						
2			Read	Read Write Execute	Read Write		Write	
3						Read Write Execute	Write	Write

Ein Prozess in Domain D_i darf *op* auf Objekt O_j ausführen, wenn *op* in der Access Matrix angegeben ist

Access Matrix (2)

Dynamic Protection

- Operationen zum Hinzufügen, Löschen von Rechten
- Spezielle Zugriffsrechte
  - Owner: spezielle Rechte des Besitzers
  - Kopieren, Verändern von Rechten
  - Transfer: Domainwechsel des Prozesses

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

Access Matrix (3)

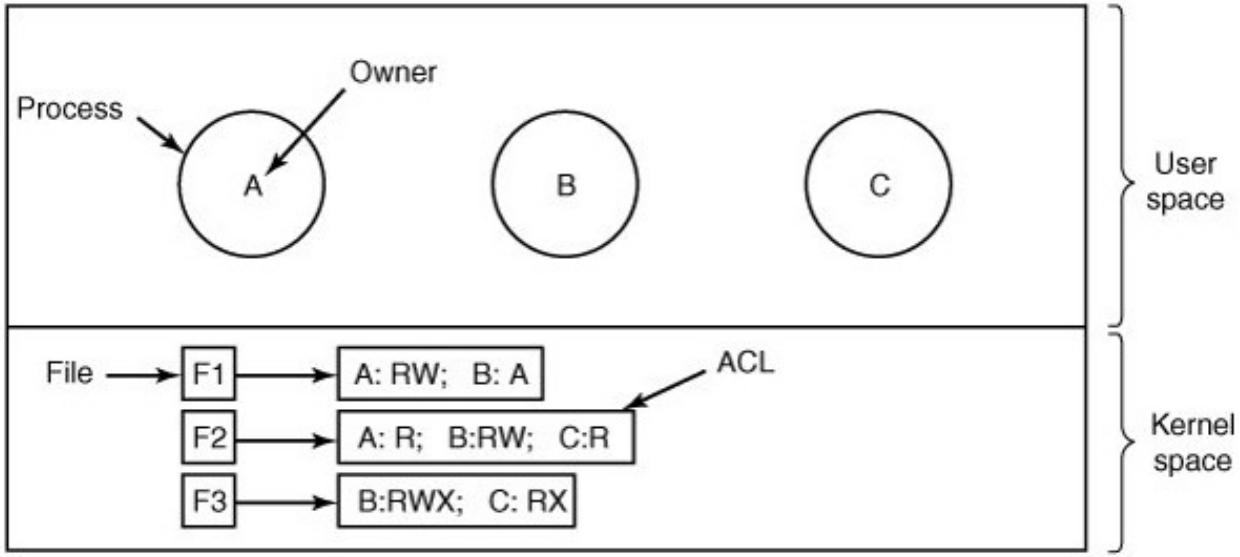
Mechanismus versus Policy

- Mechanismus
  - Betriebssystem stellt Access Matrix und Regeln zur Verfügung
  - Stellt Einhaltung der Regeln sicher
- Policy
  - Benutzer bestimmen Policy
  - Wer kann welches Objekt wie verwenden?

Access Control Lists (ACL)

Zugriffsrechte bei Objekten gespeichert (Spaltenzerlegung der Matrix)

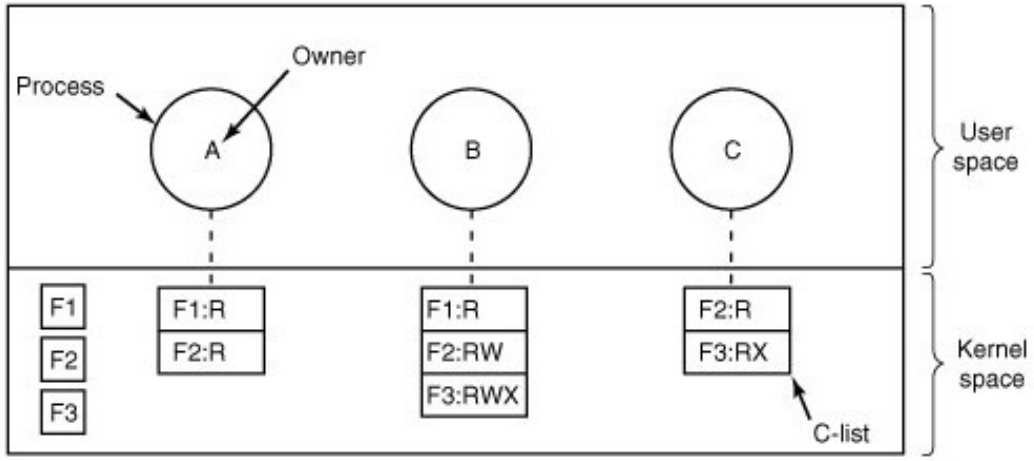
- Domain: Differenzierung nach Benutzergruppen
- leichtes Ändern der Zugriffsrechte von Objekten



Capability Lists

Für jeden Prozess gibt es eine Liste mit Objekten und erlaubten Zugriffsoperationen

- Capability Tickets regeln Zugriff auf Objekte durch Ticket-Besitzer
- Weitergabe und Vererbung von Tickets
- Fälschungssicherheit von Tickets → Verschlüsselung



Lock-Key System

- Jedes Objekt hat eine Menge von Locks (eindeutige Bitmuster)
- Jede Domain besitzt eine Menge von Keys
- Ein Prozess in D_j darf Objekt O_i zugreifen, wenn ein Key von D_j zu einem Lock von O_i passt

Bell and LaPadula's Model

- Regeln für Informationsfluss
- Hierarchie von Security Classifications (SC) für Subjects und Objects
  - z.B.: top secret, secret, public
- Operationen von Subjects auf Objects:
  - read-only (no modifications)
  - append (without reading)
  - execute (without reading or writing)
  - read-write

Security Axioms

- Simple security property
  - Read: es muss gelten SC(S) >= SC(O)
    "*no read up*"
- The *property (star property)
  - Append: SC(S) <= SC(O)
    "*no write down*"
  - Read und Write: SC(S) = SC(O)
- Subject S kann Object O_1 lesen und O_2 schreiben, wenn

$$SC(O_1) <= SC(S) <= SC(O_2)$$

Kryptographie als Security Tool

- Geschlossenes System: Betriebssystem kann Sender und Empfänger von IPC kontrollieren.
- Offenes System
  - Ohne Kryptographie kein Vertrauen auf Korrektheit von Sender/Empfänger von Nachrichten
  - Kryptographie beschränkt potentielle Sender und Empfänger von Nachrichten (deckt auch Veränderung bzw. Manipulation ab)
- Basiert auf dem Besitz geheimer Schlüssel
- Verschlüsselung, Authentifizierung

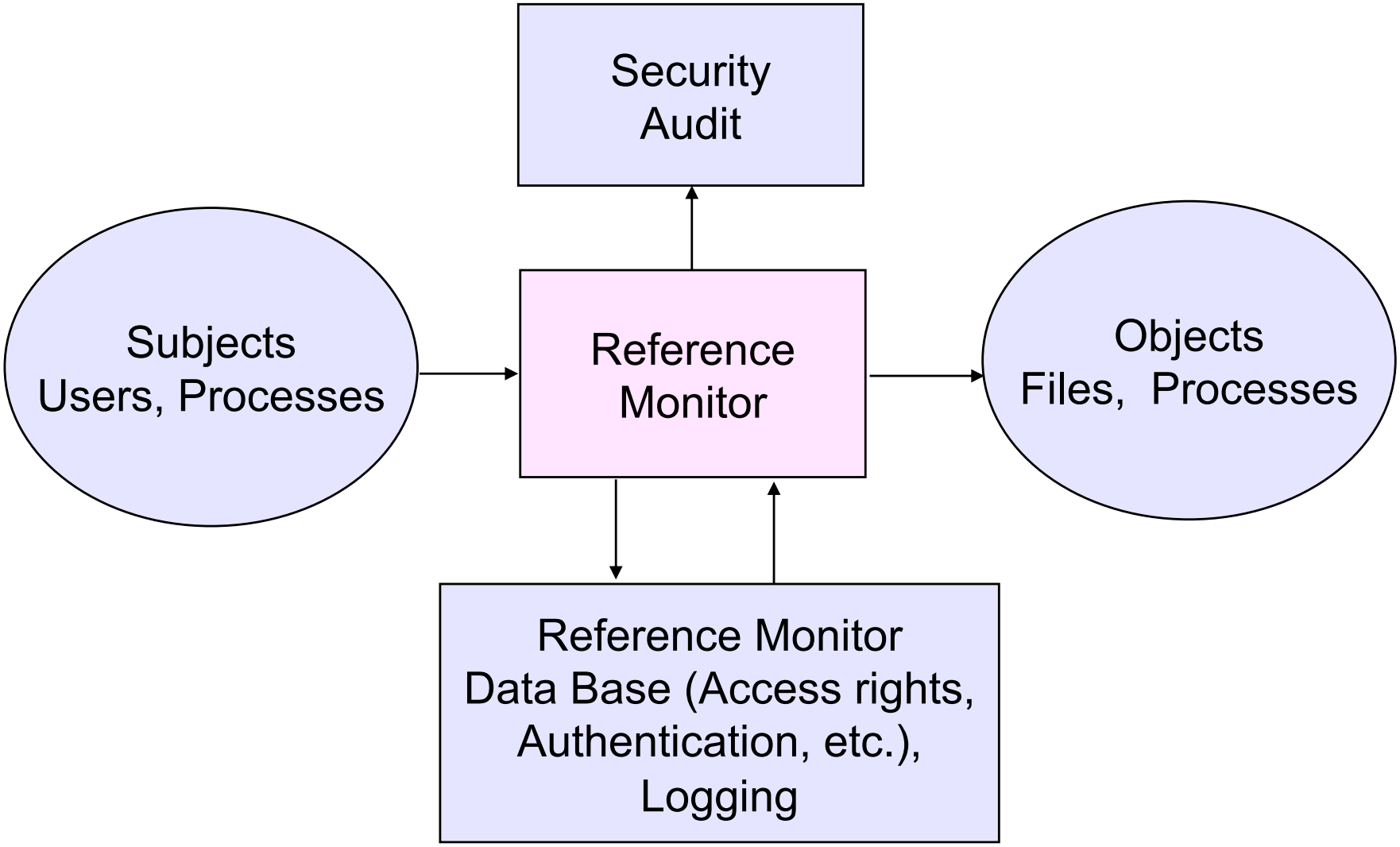
Intrusion Detection

- Threshold detection (z.B. viele Loginversuche)
- Vergleich von Zugriffsmustern mit “typischen” Zugriffsmustern von Accounts/Benutzern
- Anomaly Detection: Regeln zum Aufspüren von Anomalien

Audit records: Aufzeichnung über Operationen

  – Subject, Action, Object, Exception Conditions, Resource Usage, Timestamps

Security Architecture



Zusammenfassung

- Ziele: Confidentiality, Integrity, Availability
- Security ist nicht nur eine technische Frage
- Bedrohungen
- Design Prinzipien
- Schutzmechanismen in BS