# Rechnersicherheit

## ▼ Begriffe

### Information Security

Es geht bei Information Security um die Einhaltung von Sicherheitszielen.

### Sicherheitsziele

Die hauptsächlichen Ziele sind:

- C: **Confidentiality** → Vetraulichkeit, kein unbefugter Zugriff auf Informationen
- I: **Integrity** → Integrität, Vollständigkeit und Richtigkeit der Informationen
- A: **Availability** → Verfügbarkeit, Informationen sind verfügbar und können abgerufen werden

Weitere Ziele:

- Authenticity → Authentizität, Entität ist, was sie scheint zu sein
- Non-Repudiation → Nichtabstreitbarkeit, Auftreten eines bestimmten Events ist nachweisbar
- Reliability → Zuverlässigkeit, konsistente und intendierte Ergebnisse

Im Gegensatz dazu ist **Verbindlichkeit kein Sicherheitsziel**! Hierbei handelt es sich um einen juristischen Begriff (etwas kann verbindlich, aber nicht beweisbar sein und umgekehrt). In der Regel ist Nichtabstreitbarkeit gemeint.

# Assets (ISO 27000: 2012)

*→ Was soll geschützt werden?*

Assets sind Werte. Sie sind alles, was einen Wert für das Unternehmen hat. Sie bestimmen den Schutzbedarf der Organisation.

- Informationen
- Software
- Personen
- …

# Schutzbedarf (BSI 200-2)

*→ Wovor soll geschützt werden?*

Der Schutzbedarf wird durch **Ermittlung des größten plausibel anzunehmenden Schadens** ermittelt, der durch eine Verletzung der Sicherheitsziele entstehen kann. Man kann so Maßnahmen abschätzen und gegeneinander priorisieren.

# Schutzziel und Maßnahmen

*→ Wie wird geschützt?*

Das Schutzziel (control objective) bezeichnet, was durch implementierte Kontrollen erreicht werden soll.

Maßnahmen (controls) wirken auf das Sicherheitsrisiko ein.

# Was kann passieren? (ISO 27000)

## Event - Vorkommnis

- Umstände
- Dinge, die passieren

## Information Security Event

- Event, das sicherheitsrelevant ist
- z.B. es kommen sehr viele Netzwerkpakete an und man ist nicht sicher, ob man das managen kann → aber am Ende klappt es (daher kein Vorfall)

### Information Security Incident - Vorfall

- unerwünschte oder unvorhergesehene Information Security Events, die mit hoher Wahrscheinlichkeit zu Schaden des Business oder der Informationssicherheit führen
- schlimme Dinge, die passieren

### Attack - Angriff

- Versuch, ein Asset anzugreifen oder zu zerstören
- Unterscheid zu Vorfällen: immer mit böser Absicht (Festplatte, die zerstört wird, ist Angriff vs. Festplatte, die kaputtgeht, ist Vorfall)

## Risikomanagement (ISO 27000)

### Threat - Bedrohung

- eventuelle Ursache eines ungewollten Vorfalls, die eventuell Schaden anrichtet
- oft extern, wirkt also auf die Assets ein
- alle Dinge, die Schwachstelle ausnutzen
- kann auch Naturkatastrophe sein → Absicht nicht relevant

### Vulnerability - Schwachstelle

- Schwachstelle eines Assets, die Ziel von Bedrohungen sein kann
- oft intern, geht also von den Assets aus
- etwas im System, das ausgenutzt werden kann

### Risk - Risiko

- Potenzial von Bedrohungen, Schaden anzurichten

### Consequence - Auswirkung

- Nachwirkungen eines Events, die die Objekte betreffen

## Verteilte Systeme - “Code is Law” (Lessig)

Auf verteilte Systeme wirken verschiedene Einflüssen ein. Neben Gesetzen, gesellschaftlichen Normen und Marktlage spielt natürlich auch der Code eine Rolle. Alle sind relevant für Rechnersicherheit.

## Was ist Sicherheit?

Grob gesagt ist die Sicherheit definiert als Abwesenheit von Gefahr. Man unterscheidet zwischen Safety und Security.

### Safety

- Sicherheit des **Menschen** → in aller Regel körperliche Unversehrtheit
- Zusammenspiel aus genauen Spezifikationen, Fehlertoleranz, korrekter Implementierung und korrekter Nutzung
- der Rechner wird hier als Quelle der Bedrohung bewertet → Roboter, der Mensch angreift oder Systeme in Autos (ABS)

### Security

- Sicherheit von **Objekten**
- Zusammenspiel aus Verfügbarkeit (Schutz vor unbefugtem Vorenthalten), Integrität (Schutz vor Beschädigung oder Modifikation) und Vertraulichkeit (Schutz vor unbefugter Preisgabe von Daten)
- der Rechner wird hier als Ziel der Bedrohung bewertet → Hardware, die zerstört wird oder Software, die kompromittiert wird

## Datenschutz vs. Datensicherheit

- Datenschutz = **Schutz personenbezogener Informationen** im Sinne informationeller Selbstbestimmung
- Datensicherheit = Schutz und Sicherheit beliebiger Daten bzgl. **Verlust, Modifikation oder Vertraulichkeit**

## ▼ Social Engineering

Social Engineering ist eine Angriffsmethode, mit der soziale Interaktionen mit Menschen genutzt werden, um Ziel zu erreichen ⇒ **Methodisches Ausspähen von Infos unter Ausnutzung menschlicher Schwächen**.

- ungefähr 80% aller Angriffe nutzen Social Engineering
- z.B. Betrugsmaschen via E-Mail (Überweisung auf andere Konten) oder Herausfinden von Passwörtern
- psychologische Elemente (Vertrauen schaffen, Sicherheitsinteresse, Eile)
- auch Phishing ist eine Form von Social Engineering → Nachahmung von anderen Websites oder Anhänge in Mails

## Reverse Social Engineering

- Angreifer suggeriert dem Opfer, dass es was von dem Angreifer möchte
- man stellt Opfer ein fiktives Problem dar, auf das man eine Lösung bietet
- so wird Opfer zur gewünschten Aktion (z.B. Freigabe von Daten) verleitet

## Phishing

- Begriff meint das Angeln von Passwörtern
- Abfangen von Zugangsdaten
- irgendein Opfer

## Spearphishing

- ein ganz bestimmtes Opfer
- viel Recherche

## Vorgehensweise

1. **Informationsbeschaffung**
    - öffentliche Infos herausfinden (Social Networks, Geschäftsberichte…)
    - wichtig für Erfolg
    - Piggy Backing: Zugang mit simplen Tricks
    - Dumpster Diving: schlecht entsorgter Abfall
    - Shoulder Surfing: Über Schulter gucken
2. **Angriff**
    - Legendenbildung (Fake Story, fiktive Identität)

- funktioniert wegen Unkenntnis, fehlendem Bewusstsein und Überrumplung (oft suggerierter Zeitdruck)

3. **Auswertung**
    - sinnvolle Kombination von Teilergebnissen

## Wie verhindern?

### ISO/IEC 27032

- **Kombination** von drei Maßnahmen:
    - Verhaltensregeln, Richtlinien
    - Schulungen, Übungen
    - Technische Schutzmaßnahmen

### Unterscheidung schnelles und langsames Wissen

- bei den meisten Präventionsmaßnahmen wird das langsame (analytische) Wissen trainiert, um zu verhindern, Social Engineering-Angriffen zum Opfer zu fallen
- die Angriffe zielen aber meistens auf das schnelle (unterbewusste) Wissen ab, das bei Gefahrensituationen aktiviert wird → daher wirken sie eher selten

# ▼ Bedrohungen und Schwachstellen

## Bedrohungen

- Das “Böse”, was angreifen könnte
- Absicht, aus Versehen oder durch Umwelt
- Bewertung nach Quantität, Wahrscheinlichkeit und Ressourcen der Bedrohenden

## Schwachstelle

- Was kann schiefgehen?
- Entwurfs- und Implementierungsfehler
- kann zum Problem werden, muss sie aber nicht

- unabhängig von Auslöser (Bedroher) und Absicht
- zielt auf ein Sicherheitsziel

⇒ nur Bedrohung + Schwachstelle führen zu Problem!

## Safety

- Fault/Vulnerability: System enthält einen Fehler, es kann etwas schiefgehen
- Error/Exploit: Fehler tritt auf, etwas geht schief
- Failure: durch den Fehler verhält sich das System nicht mehr wie erwartet

## Wie geht man mit Problemen im Bereich Security um?

1. Fault Avoidance

- Fehler im Vorhinein erkennen und beseitigen
- Schwachstellen früh erkennen und beseitigen
- in Praxis eher schwer

2. Fault Tolerance

- System so gestalten, dass Fehler (Fault) nicht zu Ausfällen (Failure) führen
- Schwachstellen tolerierbar machen
- mit Schwachstellen und Exploits leben
    - Defense in Depth: Sicherheitskontrollen auf mehreren Ebenen
    - Assume Breach: immer davon ausgehen, dass Angreifer durchkommt

## Arten von Schwachstellen

1. Menschen und Prozesse
    - Social Engineering
    - Prozesse: unklare Vorgaben oder Zuständigkeiten
    - Menschen: fehlende Zeit, fehlende Kompetenz und abweichende Prioritäten
2. Softwarefehler

3. Hardwarefehler
4. Physische Probleme

# Softwarefehler

- es gibt kaum fehlerfreie Programme
- Fehler werden im Bestfall in Form von Patches behoben
- Bug ist Abweichung des realen vom intendierten Programmverhalten → entstehen bei Entwurf oder Implementierung
- in der nächsten Programmversion (release) sind die bis dahin bekannten Fehler idR behoben
- Fehlerhaftigkeit kann beurteilt werden durch:
  - Qualitätsstandard des Herstellers
  - Komplexität des Codes
  - Reife des Programmes
  - Fehlerhistorie

## Welcher Bug kann ein Sicherheitsproblem werden?

- OWASP Top 10 (typische und häufige Fehler in Webanwendungen):
  - Zugangskontrolle funktioniert nicht
  - Kryptographie klappt nicht
  - Injections/Cross-Side-Scripting
  - Unsicheres Design → es gibt Lücken, die man nicht mehr wirklich schließen kann
  - fehlerhafte Konfiguration (Misconfiguration) → Anwendung läuft z.B. auf unsicheren und unpassend eingerichteten Servern
  - Komponenten mit bekannten Sicherheitslücken werden genutzt
  - Identifikations- und Authentifikations-Fehler
  - Datenintegrität kann nicht gewährleistet werden
  - Logging- oder Monitoring-Fehler

    - Server-Side Request-Forgery (Angreifer missbraucht Serverfunktionalitäten)
- es gibt vergleichbare Listen
- Hersteller können einen informieren

### Woher weiß ich, wo die Fehler sind?

- CWE-Datenbank → Common Weakness Evaluation, in der jede Vulnerability eine ID hat
- Herstellerseiten
- Mailinglisten
- Social Media
- Zusammenfassungen

## Sicherheitslücken

### Buffer Overflow

- eines der häufigsten Probleme
- ermöglicht das Ausführen von eigenem Code oder zumindest das Ändern von Dateien
- man hat einen Puffer (in der Regel ein Array bzw. eine Zeichenkette)
- man kann eigene Instruktionen einschleusen oder eine Funktion aufrufen, indem man die Instruktionen bzw. die Rücksprungadresse auf den Stack schreibt
- man kann auch Global Object Table oder Funktionspointer umbiegen
- treten im Stack und im Heap auf, wenn Puffergrenzen nicht geprüft werden
- können zu Änderungen für Codeausführung, Änderungen an Daten oder Lesen von Daten führen

**Gegenmaßnahmen**

- Grenzen prüfen
- Canaries um wichtige Daten platzieren → vor Zugriff auf z.B. Funktionspointer oder Rücksprungadresse wird Canary-Wert geprüft

    - Canary-Wert ist ein int im Stack vorm Return-Pointer, dessen Wert bei Programmstart zufällig gewählt wird → um Return-Pointer zu überschreiben, muss man auch Canary überschreiben (und der Wert wird zwischendrin gecheckt, sodass BO auffällt)
- Adressen zufällig wählen (ASLR) → Adress Space Layout Randomization
    - Module werden in zufällige Speicherblöcke geladen → kein Erraten von Position von Prozessen
    - nicht für alle Daten möglich
    - teilweise zu kleiner Zufallsraum
- Stack und Heap nicht ausführbar machen
    - Data Execution Prevention: Bereiche des Arbeitsspeichers werden als “nur für Daten” markiert → dort darf kein Code ausgeführt werden
    - NX-Bit: Technik von x86-Prozessoren, die ein bestimmtes Bit für den Stack und Datenbereiche gesetzt wird, sodass man diese nicht mehr ausführen kann

## Return Oriented Programming (ROP)

- Voraussetzung:
    - kein eigener ausführbarer Code hochladbar
    - Stack mit Rücksprungadressen manipulierbar
    - CPU unterscheidet nicht zwischen normalem Sprung und Rücksprung
- Idee: man sucht Codeschnipsel (Gadgets) im bestehenden Code, die was sinnvolles machen, und setzt aus diesen Fragmenten ein Programm zusammen
- künstlichen Stack so aufbauen, dass Rücksprungadresse jeweils zum nächsten Fragment zeigt
- kein Schutz durch NX-Bit, da der Code sich im als ausführbar markierten Speicher befindet

## Integer Underflow (analog Overflow)

- normalerweise ist x-y < x (für y > 0)

- bei Binärzahlen ist das anders → aufgrund der begrenzten Stellenzahl können unter Umständen Ergebnisse nicht richtig dargestellt werden
- Statusbits der CPU sind zwar gesetzt (Overflowbit), müssen aber überprüft werden → das ist nicht in allen Programmen der Fall
- treten auf bei
    - Überlauf/Unterlauf des Wertebereichs
    - Zuweisung von Typen mit/ohne Vorzeichen
    - unterschiedliche Größen (Byte, int, long)

# Sprachgrenzen und Injections

Grundlage: Die meisten Systeme bestehen aus verschiedenen Komponenten, von denen jede eine eigene Sprache hat.

**Beispiel Webanwendung**

- Browser spricht mit Server über HTTP
- Webserver liefert Webseiten in HTML aus, die JS enthalten können
- Webserver führt Python aus
    - Python ruft Programme über Shell auf
    - Python spricht mit Datenbank in SQL

→ an jeder Sprachgrenze kann es zu Missverständnissen kommen, bereits geprüfte Daten können in der anderen Sprache bösartig sein

- man sollte für Zielsprache spezifisch encoden

## SQL Injection

- Userinput, der restlichen SQL-Code quasi ausschaltet

## Path Traversal

- System soll eigentlich Daten aus einem Verzeichnis verarbeiten
- Eingabe eines Verzeichnispfades wie “../../../etc/passwd” würde Bedingung verletzen
- Lösung: String normalisieren → man rechnet Ergebnis von Vorgabe und Eingabe aus und checkt, ob diese Anfrage erlaubt wäre

Aktuelles Beispiel Log4j

- Logmeldungen können Platzhalter enthalten → diese kann man mit JNDI.Aufrufen füllen, die externe Ressourcen einbinden oder Klassendefinition nachladen
- so kann es zu Remote Code Execution kommen (Ausführen von unerwünschtem Code auf einem entfernten Rechner)

## Crosssidescripting: XSS/CRSF

- persistentes XSS: vom Angreifer kontrolliertes HTML wird eingebettet und ist für andere Nutzer sichtbar
  - Lösungen:
    - Zielsystem muss wissen, was was ist → was bekommt man gerade, als was soll der Code interpretiert werden?
    - auf kritische Zeichen in Zielsprache testen
- nicht-persistentes XSS: Angreifer-Code nur für direkten Aufrufer eingebettet (zB Suchbegriff auf Ergebnisseite)
  - Lösung: wie bei nicht-persistentem XSS
- CSRF (Cross Side Request Forgery): auf anderer Website ist Link eingebettet und Browser ruft dann die Seite auf und ruft eine Funktion bei uns auf oder sendet eine HTTP-Request
  - vor allem problematisch, wenn Nutzer eingeloggt sind
  - Lösung: Information einbauen, die Angreifer nicht kennen kann
    - z.B. Zufallszahl, die nicht als Cookie gespeichert wird

## TOCTOU → Time of Check vs. Time of Use

- ähnlich zum kritischen Abschnitt → eine Eigenschaft wird erst geprüft und später wird basierend auf Prüfung gearbeitet
- Beispiele:
  - prüfen ob Datei vorhanden, wenn nicht beschreiben
  - prüfen, wo ein SymLink (Referenz auf Zieldatei) hinführt, dann nutzen
  - prüfen, ob noch genug Speicher

- Zeit zwischen TOC und TOU wird genutzt um den Zustand des geprüften Elementes zu verändern und das Prüfergebnis für den Programmverlauf irrelevant zu machen
- z.B. Virencheck ohne Ergebnis, dann wird Virus eingeschleust, aber System geht noch davon aus, dass kein Virus da ist

# Mobile Code

- man möchte fremden Code ausführen → als Client vom Server, .exe-Dateien, curl…
- Risiken:
  - Code wird mit Rechten des Nutzers ausgeführt
  - Betriebssystem bietet keinen Schutz, da reguläre Aktion des Nutzers
  - Vertrauen in Quelle und Programmierer

**Klassisches Modell**

- Code lokal installieren und dann nutzen
- Code unter der Kontrolle des Admins

**Mobiler Code**

- Code wird kurz vor Ausführung übertragen und kann sich jederzeit ändern
- Code unter Kontrolle des Senders

## Ausführbare Dateien

- klassische ausführbare Dateien werden aus Internet geladen und ausgeführt (z.B. mit curl)
- werden mit Rechten des Nutzers ausgeführt → OS bietet keinen Schutz

## Plugins/Extensions

- keine eigenständigen Anwendungen, sondern laufen im Kontext von Anwendung
- können alles machen, was Anwendung darf
- Risiken:
  - gehen häufig am Patchmanagement vorbei

    - können meist mit geringen Rechten nachinstalliert werden
    - können von Dritten kommen

### Sandbox

- Idee: fremder Code darf ausgeführt werden, aber darf kaum mit System interagieren → nur unkritische Funktionen werden angeboten
- isolierte Umgebung zur geschützten Ausführung von Software
- Probleme:
    - Was sind unkritische Funktionen?
    - Was muss verboten sein?
    - Wie abschotten?

### Same-Origin-Policy

- kommunizieren ja, aber nicht mit jedem Server
- Lösung:
    - Mobiler Code darf nur mit Server kommunizieren, von dem er geladen wurde
    - Ursprungsserver liefert Whitelist der möglichen Kommunikationspartner (z.B. CSP (Content Security Policy)
    - Zielserver liefert Whitelist, welche Skripte kommunizieren dürfen (z.B. CORS (Cross Origin Resource Sharing) → Bereitstellung von Inhalten aus einer anderen Quelle)
- Nachteile:
    - gibt Serverarchitektur vor
    - verleitet zu Tricks zum Umgehen
    - muss gut umgesetzt sein

## Weitere Bugs

- Autorisierung nicht für jede Operation geprüft
- zu lasche (nicht umgesetzt) oder zu scharfe (nicht genutzt) Sicherheitsfunktionen

- Kryptoalgorithmen fehlerhaft umgesetzt
- Debug-Funktionen noch verfügbar
- Komponenten enthalten Fehler
- Daten werden an unsicheren Orten abgelegt (z.B. temp-Ordner)

---

# Angriffe von Außen

## Malware

- Programme, die speziell dafür entwickelt wurden, Sicherheitsziele zu verletzen
- Viren, Trojaner, Wurm, Bots, Remote Access Toolkits, Logische Bomben…

## Logische Bombe

- Programm führt Verhalten solange korrekt aus, wie eine Bedingung erfüllt ist
- wird Bedingung verletzt, wird eine bereits enthaltene Schadsoftware ausgeführt

## Falltür

- Programm funktioniert, aber es gibt eine geheime Umgehung der Zugangs- oder Zugriffskontrolle, um unautorisiert privilegierte Funktionen auszuführen
- vom Programmierer absichtlich eingebaut

## Trojaner

- Programm, das sinnvolle Nutzung vorgibt, aber auch Schadfunktionen erhält
- kann in ausführbarer Datei, Installationsskript, Shellskript enthalten sein
- Gegenmaßnahmen:
  - Software aus vertrauenswürdiger Quelle
  - vor Installation prüfen
  - nicht installierte/zentral frei gegeben Software verbieten

# Virus

- Schadsoftware, die sich viral weiter verbreiten möchte → haben Reproduktionsteil
- Befehlsfolgen, die Kopien von sich selbst in anderen Speicherbereichen ablegen
- Zielbereiche: Programme im Hauptspeicher, Programmdateien, Konfigurationsdateien, Bootsektoren
- Kennungen müssen eingebaut werden, damit Viren sich nicht selbst wieder infizieren
- Schalen-Virus: enthält Aufruf an ursprüngliches Programm, aber erst startet Virus → zwei Dateien, wobei ursprüngliches Programm als internes Unterporgramm aufgerufen wird
- additiver Virus: kopiert sich in Datei rein, indem Sprung zu sich selbst eingebaut wird → Datei wird größer, Startadresse zunächst auf Viruscode
- ersetzender Virus: Virus ersetzt Programm und schreibt sich rein → Programm wird nicht mehr funktionieren

## Generationen von Viren

1. einfache Viren → nur Replikation ohne zusätzliche Maßnahmen
2. Selbsterkennung → erkennt, ob Programm schon infiziert
3. Tarnkappen-Viren → durch Gegenmaßnahmen Entdeckung verhindern
4. Selbstmodifizierung → Modifikation bei Replikation, um Erkennung zu erschweren
5. Polymorphie → selbstmutierend, die Programm z.B. mit verschlüsselter Version von sich selbst infizieren

# Wurm

- lauffähiges Programm, das sich über Netz von Rechnern vervielfältigen kann
- verteiltes Programm, das selbstständig in andere Rechner eindringt und sich dort einnistet und von dort aus neue Ziele sucht
- verbreitet sich im Gegensatz zum Virus, ohne fremde Dateien mit Code zu infizieren

## Bot

- System, das der Angreifer aus Ferne für weitere Angriffe nutzen kann
- idR durch Command and Control Server gesteuert
- Ziele:
  - Weiterverbreitung
  - Fluten (DDoS)
  - Spam
  - Informationen exfiltrieren
  - Transaktionen manipulieren
- Bots, die primär für interne Ziele gedacht sind, werden Remote Access Toolkits (RAT) genannt

## Denial of Service (DoS)

- Programme, die System durch übermäßigen Ressourcenverbrauch lahmlegen
- innerhalb des Systems: starten ständiger Kopien von sich selbst (Verbrauch von Hauptspeicher, CPU)
- auf andere Systeme: häufiger gezielter Aufruf von ressourcenhungrigen Aktionen, sehr große Zahl einfacher Aufrufen
- DDoS: Verschleierung der Herkunft durch viele infizierte Systeme
- Reflection: Angriff durch fehlgeleitete Antworten

## Ransomware

- Hauptfunktion von Malware
- Nutzer wird erpresst, um Schadfunktion zu vermeiden oder Schaden rückgängig zu machen
- Arten:
  - Rechner sperren → kaum mehr nutzbar
  - Verschlüsselung → am häufigsten, Daten nicht nutzbar (häufig auch mit Rechteausweitung und Löschen von Backups)

    - Veröffentlichung
    - DDoS → Drohung mit DDoS-Angriff
- Hauptursache von Schäden in den letzten Jahren -> Millionenumsätze und Professionalisierung

# Spam

- Malware kann genutzt werden, um unerwünscht Massenmails zu versenden
- hohe Kosten durch Spam: Arbeitszeit zum Entfernen, Spam-Filter, Belegung von Ressourcen
- Gefahr für infizierte Systeme: System oder ganzes Netz kann auf Blacklist kommen (andere Server nehmen keine Mails mehr an) oder Mailadressen werden aus dem System extrahiert

# Gegenmaßnahmen

## Scanner

- Programm, das Speicherblöcke liest und prüft, ob Pattern auftritt
- Bitmuster ist charakteristisch für jeweilige Malware
- kann aber nur bekannte Malware finden
- Zeitverzug zwischen Virus und Erkennung
- häufiger Fehlalarm

## Activity Monitor

- Systemprogramme oder Netzwerkkomponenten, die alle Aktivitäten überwachen
- Alarm bei verdächtigem Verhalten
- selten wirklich selbstlernend

## Integrity Checker

- Programme, die für Programm- und Konfigurationsdateien Prüfsummen berechnen
- Änderungen können durch Veränderung der Prüfsumme erkannt werden

## Hardware

- auch Hardware kann Fehler verursachen:
- vor allem geht Hardware nicht mehr ohne Software → Firmware kann Softwarefehler enthalten
- Änderungen in der Hardware können Annahmen in der Software verändern
  - schnellere/langsamere Hardware
  - DMA (Direct Memory Access)-Zugang
  - Multicore
  - Virtualisierung bei Cloud Computing
- Hardware-Entwicklung kann Annahmen verändern → Timing-Angriffe, Bruteforce wird realistisch
- Hardware selbst kann echte Fehler haben

## Physische Sicherheit

- Gefahren für Sicherheit:
  - Diebstahl und Verlust
  - Manipulation der Hardware (Keylogger, Netzwerksniffer)
  - Zugriff auf ungesicherte Dokumente
  - Abhören und Mitschneiden
  - Vandalismus
- Maßnahmen:
  - Zugang zum Gebäude und den Räumen regeln
  - Prüfungen auf Manipulation

## Umwelteinflüsse

- Gefahren:
  - Strom (Ausfall, Überspannung)
  - Feuer (Rauch, Löschwasser)
  - Temperatur

- Erdbeben, Explosionen
- Großveranstaltungen

# ▼ Datensicherung

## Schutzziel

- soll verhindern, dass “Daten verloren gehen”
- wichtig vor allem für Integrität und Verfügbarkeit
- aber auch für Authentizität und Nichtabstreitbarkeit

## Wogegen will man sich schützen?

- Benutzerfehler
- Fehler des Betriebspersonals
- Hard- oder Softwarefehler
- Vandalismus durch Hacker oder Viren
- Diebstahl
- (Natur-)Katastrophen

## Mögliche Lösungen

- Transparente Redundanz
  - Fehler automatisch ausgeglichen
  - zeitnah und meist ohne Verlust
  - hilft nur gegen Fehler in unteren Schichten
  - meist weiterhin ein Single-Point-of-Failure (RAID-Controller, Failover-Komponente)
  - zusätzliche Komplexität
  - kein Schutz vor Diebstahl ohne räumliche Trennung
  - Fehleingaben werden umgesetzt
  - Nutzer merkt Sicherung nicht

    - z.B. Raid
- Sicherungskopien (Backup)
    - Fehler werden später behoben
    - meist nur diskrete Sicherungen
    - auch gegen Fehler in höheren Schichten und Nutzerfehlbedienungen
    - Gedanken über Sicherungsstrategien nötig → muss diskretisiert weren

## RAID - Transparente Redundanz

- mehrere Festplatten zusammenschalten
- bei Ausfall Rückgriff auf andere Festplatten
- Failover: zwei vollständige Systeme, im Fehlerfall wird anderes genommen
    1. beide Systeme sind vollständig synchron
    2. Hot-Standby: Zweitsystem läuft immer und pflegt Datenänderungen nach
    3. Cold-Standby: Zweitsystem wird erst im Fehlerfall hochgefahren

## Sicherungskopien

- System kann wieder in den Zustand zu einem anderen Zeitpunkt zurückversetzt werden
- Frage der Wahl des Zeitpunkts
- Arten:
    - day-zero backup: Sicherung des Anfangssystems
    - full backup: Kopie jeder einzelnen Datei
    - differential backup: eine Kopie jeder Datei, die sich seit Komplettsicherung geändert hat oder die hinzukam
    - incremental backup: eine Kopie jeder Datei, die sich seit Sicherung geändert hat oder die hinzukam

## Sicherungsmedien

- typischerweise auf Tapes
- Onlinespeicher in Clouds

- Festplatten
- optische Medien (DVD, BluRay)

# ▼ Identifikation und Authentifikation

## Identifikation

- man möchte sich bei einem Rechner identifizieren
- je nach Rolle beim Rechner sind verschiedene Funktionen möglich und diese sollen nachvollziehbar zugeordnet werden
- hierfür gibt es interne Repräsentationen von externen Benutzern → man unterscheidet Rollen und keine Individuen

## Authentisierung

- Benutzer müssen sicher sein, mit richtigem System zu arbeiten
- man muss Nachweis bringen, dass man in aktueller Rolle agieren darf

⇒ Dreischritt: Wer bin ich? (Identifikation) → Bist du wirklich du? (Authentisierung) → Darf ich hier sein? (Autorisierung)

## Authentisierung von Personen

- Was man ist? → spezifische Merkmale
  - Biometrie
  - man braucht Tradeoff zwischen falscher Akzeptanz und falscher Ablehnung → eher höhere falsche Ablehnung als falsche Akzeptanz
  - Eigenschaften nur fehlerbehaftet messbar (ungenaue Messtechnik)
  - Fälschungen (Ausnutzen konkreter Messtechnik/ungenauer Prüfung)
- Was man hat? → ausgehändigte Gegenstände
  - Klassische Lösung der Offline-Welt
  - Analog für Rechner: Codekarten, Chipkarten, USB-Sticks, Schlüssel für Terminals
  - Smartcards enthalten einen vollständigen Rechner mit eigenem OS und Anwendungen

        - Schnittstelle zu anderen Systemen
        - Idee: Schlüssel bleiben in Karte, alle Berechnungen dort
- Was man weiß? → Vereinbarte Fakten oder Algorithmen
    - aktuell am häufigsten eingesetzte Variante
    - Passwortverfahren
    - Frage- und Antwortverfahren: z.B. TAN-Liste
    - Durchführung eines Algorithmus f: System nennt Zahl, Benutzer errechnet f(x)
- Wo man ist? → Ort
    - physischer Ort: mobiles Gerät funktioniert nur auf dem Campus
    - logischer Ort: jeder der Im TU-Netz ist, darf auf Bibliothekssysteme zugreifen
    - selten alleiniges Verfahren, aber häufig als zusätzliche Authentifizierungsschicht

## Passwörter

- zwischen dem Nutzer und dem System vereinbarte Zeichenkette
- wiederverwendbare Passwörter können für mehr als einen Authentifizierungsvorgang genutzt werden
- gespeichert in Benutzerdatenbank oder Passworttabelle:
    - geschützt gegen unbefugtes Schreiben
    - Schutz gegen unbefugtes Lesen
        - Leseschutz → Zugriff nur für Admins
        - gehashtes Abspeichern → nur Ergebnis wird gespeichert

## Passwort-Hash und Salt

- Basis ist kryptographische Hash-Funktion
- gleiche Passwörter haben gleichen Hash → Vorberechnung von Hashes für typische Passwörter
- Ergänzung: EInsatz von Salt

    - zusätzlicher Wert wird in Berechnung eingezogen
    - beim Setzen des Passworts zufällig bestimmt, mit dem Passworthash abgespeichert
    - gut, wenn mehrere Nutzer dasselbe Passwort haben
    - streut zusätzliche Komplexität ein → Berechnung bei Brute-Force wird verlangsamt

### Moderne Passwort-Hash-Funktionen

- $1$ MD5 → mehrfache Nutzung von MD5, Passwort und Salt jönnen beliebig viele Zeichen haben
- $2$/§2y$ Blowfish
- $5$/$6$
- man kann im laufenden Betrieb Passworthashverfahren ändern
- wenn User Passwort verändert, wird es als $2$ usw. gespeichert

## Wörterbuchangriffe

- Passwörter sind nicht gleichverteilt:
    - häufig natürliche Wörter oder Wortkombinationen daraus
    - meist aus Umfeld des Nutzers
    - generierte Passwörter haben Akzeptanzprobleme oder werden aufgeschrieben
- Angriffsmöglichkeit: Liste typischer Passwörter ausprobieren und mit typischen Änderungen kombinieren
- Rainbow-Tables als Kompromiss

## Rainbow-Tables

- schnelle, speichereffiziente Suche nach ursprünglichen Zeichenfolge für gegebenen Hash
- Reduktionsfunktion wandelt Hashwert in Plaintext um (nicht enthasht, sondern beliebige Umwandlung) → aus diesem Text neuen Hash (mehrfach), sodass Hashkette entsteht

## Allgemeine Passwortprobleme

- Weitergabe
- Erraten
- Über-die-Schulter-schauen
- Social Engineering
- Default-, Initialwert
- Passwort-Cracker
- gleiches Passwort

## Einmal-Passwörter

- Bekanntwerden eines Passworts ist evtl. dann hinnehmbar, wenn es eh nur ein mal gültig ist
- beruhen auf gemeinsamen Geheimnis zwischen Benutzer und System
- zeitbasiert: basierend auf Zeit und gemeinsamen Geheimnis wird Code berechnet → hinreichend synchrone Uhren benötigt
- Challenge-Response-Verfahren: System generiert Zahl (challenge), die zusammen mit gemeinsamen Geheimnis in eine Funktion eingeht (response) → z.B. Onlinebanking mit Flickercode
- Code-Book: Tabelle mit Einmalpasswörtern oder Funktion, die Reihe von Passwörtern erzeugt → z.B. TAN-Listen

## Zwei-Faktor-Authentifizierung

- Ziel: Nachteile einzelner Techniken ausgleichen
- zwei verschiedene Grundmethoden kombinieren, die getrennte Faktoren abarbeiten → nicht Besitz generiert Wissen, sondern Besitz und Wissen
- Nutzung und Verwaltung schwieriger als einzelner Faktor
- typische Fehler:
  - Fallback vorhanden: ein Faktor reicht
  - beide Faktoren auf gleichem Weg gefährdet (Banking-App = TAN-App)
  - ein Faktor wird vernachlässigt (schlechtes Passwort)

# Authentisierung durch vertrauenswürdige Dritte

- Client und Server müssen erstmalig kontaktieren, es gibt noch keinen geheimen gemeinsamen Schlüssel
- Vertrauensinstanz
- bekannte Verfahren:
  - Needham und Schroeder
  - KryptoKnight
  - Kerberos
  - OpenID
  - oAuth
- aufwändiges Protokoll aus mehreren Schichten

## Needham-Schroeder

- A und B wollen kommunizieren
- Authentisierungsdienst AD hat Schlüssel von A und Schlüssel von B mit A und B ausgetauscht
- A sendet unverschlüsselte Nachricht an AD, die eigene Identität und Identität des gewünschten Kommunikationspartners sowie Zufallszahl Nonce Na beinhaltet
- AD sendet Antwort an A, die mit As Schlüssel verschlüsselt ist und die auch die Nonce Na beinhaltet, außerdem beinhaltet sie den Sitzungsschlüssel, die Identität von B und ein Block, der mit Bs Schlüssel verschlüsselt ist und der die Identität von A und den Sitzungsschlüssel beinhaltet
- A sendet den Teil, der mit Bs Schlüssel verschlüsselt ist, an B  => jetzt haben beide Sitzungsschlüssel

## Kerberos

- Ziel: Authentisierung in großen verteilten Server-Client-Systemen, in denen man damit rechnen muss, dass
  - Benutzer sich für wen anders ausgeben
  - Maschinen eine falsche Identität haben

    - Nachrichten abgefangen und später für Zugang eingespielt werden
- weitere Ziele:
    - Transparenz: Nutzer sollte außer Passworteingabe nichts vom Authentisierungsmechanismus mitbekommen
    - Skalierbarkeit: System auch bei tausenden Maschinen noch arbeitsbereit
- Entkopplung von Anmeldeserver, Authentifizierung für Dienstnutzung und eigentliche Dienstnutzung
- Client meldet sich einmalig am Authentisierungsserver AS (einzige Maschine mit Passwortdatenbbank) an und erhält Sitzungsticket (mit Passwort des Clients verschlüsselt)
- Client erhält von Ticket Grantig Server Tickets für einzelne Dienste (dafür authentifiziert er sich mit Sitzungsticket)
- Client nutzt dann Dienst mit Service-Ticket
- Tickets sind jeweils verschlüsselte Nachrichten → nur berechtigter Server kann Ticket öffnen
- Kerberos-Systeme können auch zusammenarbeiten
- Vorteil: AS sehr klein, viel Traffic hat man eher mit TGS

### Verfahren, die auf HTTP aufsetzen

- Authentisierungsprotokolle für Webanwendungen
    - OpenID/OAuth
        - Benutzer gibt OpenID-Provider beim Dienst an
        - Browser wird zum OpenID-Provider weitergeleitet
        - Nutzer meldet sich dort an
    - Liberty Alliance
    - Google SignIn
    - Login with Amazon
- Nachteil: Token lange gültig, keine “richtige” Abmeldung möglich
    - Lösung: häufigeren Login fordern (kürzere Time to Live)

# ▼ Zugriffskontrolle (Autorisierung)

## Grundmodell Zugriffssystem

- Ein Tripel (s,p,o) heißt Zugriffsweg, wobei s ein Subjekt, p ein Operator und o ein Objekt ist
- Menge der Subjekte S: Nutzer, Prozesse, Programme, Rechner, Geräte
- Menge der Objekte O: Geräte, Dateien, Speichersegmente, Prozesse, Funktionen
- Menge der Operatoren P: meist typspezifisch
  - Beispiele bei Dateien: Erstellen, Schreiben, Lesen, Verschieben, Löschen
  - Beispiele für Funktion: Aufrufen, Ersetzen, Schreiben

## Gruppen

- man kann Subjekte, Objekte und Operatoren in Gruppen zusammenfassen
- sie können disjunkt, überlappend oder geschachtelt sein
- Vereinfachung

## Rollen

- Rollen sind spezielle Form von Gruppenbildung
- konkretes Aufgabenprofil mit dazugehörigen Rechten
- Subjekte könne aktive Rolle wechseln (Sitzung verändern) und auch mehrere Rollen haben

## Zugriffsrelationen

- Darstellung der erlaubten Zugriffswege:
  - verbotene Zugriffswege aufzählen (default permit)
  - erlaubte Zugriffswege aufzählen (default deny) → sicherer
  - in Praxis auch Mischformen
- erlaubte Zugriffswege werden in Zugriffsrelation zusammengefasst
- kann man als Matrix schreiben

### Speicherung der Zugriffsrelation

- zentral: Tripel zusammengefasst aufgelistet
- subjektorientiert: Tupel pro Subjekt aufgelistet
- objektorientiert (Zugriffskontrolllisten): Tupel pro Objekt aufgelistet
- operatororientiert: Tupel pro Operator aufgelistet

## Capabilities → Beispiel für subjektorientiert

- jedes Subjekt erhält Objektverweis, der neben dem Bezeichner auch die Rechte enthält
- “Pointer mit Rechten dran”, können weitergegeben werden (Link von Google Docs)
- können kopiert oder als Parameter übergeben werden
- müssen gegen Manipulation geschützt sein und bei jedem Zugriff geprüft werden
- Nachteil: können weitergegeben werden
- Vorteil: super schnell

## Zugriffskontrolllisten (ACL) → Beispiel für objektorientiert

- an jedem Objekt gibt es eine Liste der Subjekte und ihrer Rechte
- Liste linear bei jedem Zugriff durchlaufen (bei Capabilities nicht nötig, schneller)
- Vergabe und Entzug einfach (bei Capabilities schwer)
- aktueller Stand der Zugriffsrelation kann jederzeit erstellt werden (bei Capabilities schwer)
- besonderer Schutz bei ACL nötig, aber sie sind außerhalb des direkten Einflussbereichs von Subjekten (bei Capabilities schwieriger)

⇒ Praxis häufig Kompromiss zwischen ACL und Capabilities

## Lock-Key-Systeme

- jedes Subjekt hat Capability-Liste mit Einträgen (K,o,r), wobei K ein Schlüssel ist und o ein Objekt und r ein Recht

- jedes Objekt hat ACL mit Einträgen (L, a), wobei L ein Schloss und a eine Menge von Rechten ist
- ein Zugriff eines Subjektes s auf ein Objekt o mit Recht r ist genau dann zulässig, wenn s eine Capability (K,o,r) repräsentiert, für die es ein passendes Schloss gibt → L = K und r in a

**Beispiel Unix**

- alle Rechte einer Datei werden anhand des Rechtefeldes im Inode einer Datei geprüft
- daraufhin wird Dateideskriptor erzeugt, der zum Indizieren einer prozessspezifischen Tabelle offener Daten verwendet wird
- von solcher Tabelle führt Eintrag in systemweite Dateitabelle mit spezifischen Rechten

Beispiel NT

- Windows NT ist ähnlich organisiert: jedes Objekt hat ACL mit mehrere Einträgen

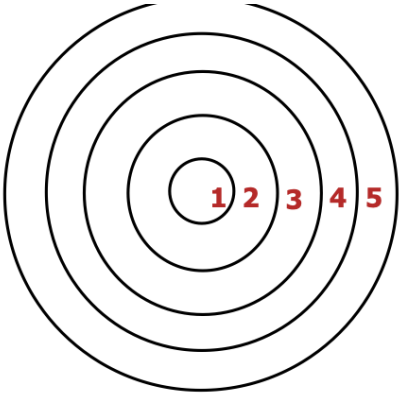
# Ringschutz

- Objekte mit abgestufter Schutzwürdigkeit bzw. abgestuften Rechten können in Ringen dargestellt werden
- mit Ringen könnte man auch Ausführungsumgebungen ineinander schachteln (OS Kern (1) auf dem OS Kern (2) auf dem Anwendung (3) läuft usw.)

**Mögliche Verwendung:**

Subjekt der Klasse (Ringnummer) $k$ darf auf Objekt der Klasse $j$ zugreifen, gdw $k \leq j$.

Objekte mit Ringnummer $i$ dürfen nur von Subjekten mit Ringnummer $k \leq i$ benutzt werden.






# Speicherschutz: Speichertrennregister

- im Einbenutzerbetrieb reicht es aus, OS gegen Fehler der Anwendungsprogramme zu schützen
- wenn man OS an Rand des Speichers legt (Anfang oder Ende), muss man nur eine Adresse haben, die kleinste/größte zulässige Adresse für Anwendungsprogramme festlegt
- Hardware muss Einhaltung gewährleisten

## Speicherschutz: Speichergrenzregister

- im Mehrprogrammbetrieb und mit relativer Adressierung hat Programm zusammenhängenden Speicherbereich, dessen Anfang und Ende man überprüfen kann
- zur relativen Programmadresse wird Inhalt des Basisregisters hinzugezählt → also keine kleineren Adressen als Basisadresse
- Obergrenze wird mit Grenzregister geprüft
- bei Überschreitung löst Prozessor Unterbrechung aus

## Speicherschutz: Gestreute Adressierung

- Tabellenbasisregister und Segmenttabelleneinträge werden um Längenfeld ergänzt
- bei Überschreitung Unterbrechung

## Rechtevergabe

- Vergabe und Entzug von Rechten sind in dynamischen Systemen nötig
- da beides Operationen an der Zugriffsrelation sind, muss Nutzung kontrolliert werden
- Vergaberelation legt fest, welche Subjekte welche Einträge verändern dürfen

## HRU-Modell

- kann man feststellen, ob Rechtevergabesystem sicher ist?
- Voraussetzungen:
  - Ausgangszustand: aktuelle Zugriffsrelation und Mengen S, P, O

- Definition sicher: bzgl. einem Recht $r \notin ZR$: keine Folge von Zustandsänderungen, sodass $r \in ZR$
- Definition Zustandsübergänge: Kommandos, die aus Elementaroperationen programmiert werden (delete, enter, createsubject…)

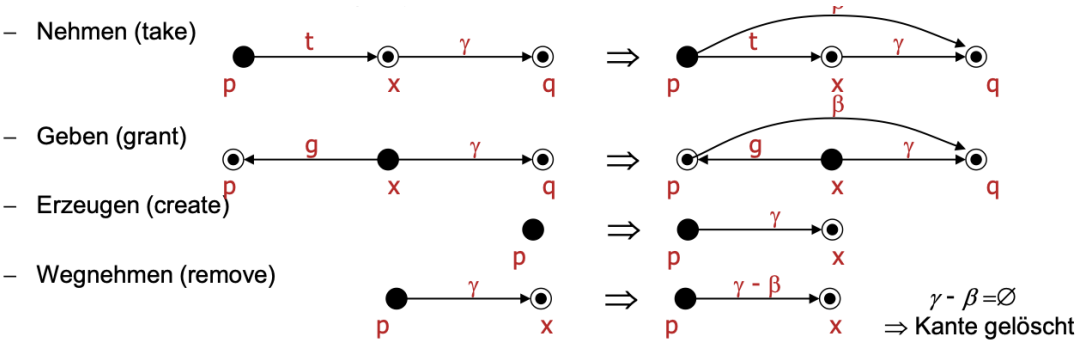
- Ergebnis:
  - ohne Einschränkungen unentscheidbar → Halteproblem!
  - nur eine Elementaroperation pro Kommando: entscheidbar, aber NP-vollständig

# Take-Grant-Modell

- vereinfachtes Modell, das nur bestimmte Rechte betrachtet und vorgegebene Zustandsübergänge definiert
- Modell: Graph mit gerichteten und bewerteten Kanten → Subjekte und Objekte sind Knoten
- Kante mit Label l von Knoten A zu B bedeutet: A hat Recht l an B
- Graph stellt ZR und die Menge der Subjekte und Objekte dar

## De-jure-Analyse

- Kann Subjekt s an das Recht p an Objekt o gelangen?
- vier mögliche Übergänge:

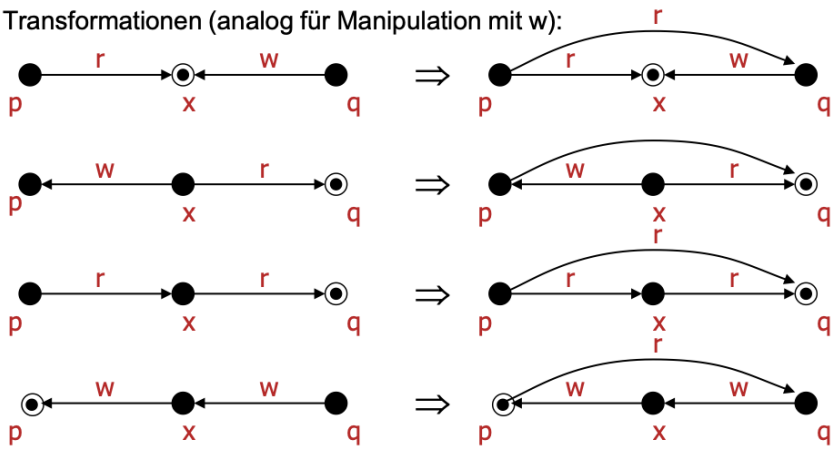



- Frage kann in linearer Zeit beantwortet werden

## De-facto-Analyse

- Kann Subjekt s ggf. über Umwege an die im Objekt o gespeicherte Information gelangen oder diese manipulieren?
- trivial, wenn direkte r oder w Kante zwischen s und o
- vier mögliche Transformationen:

– Wieder vier mögliche Transformationen (analog für Manipulation mit w):
  – „Present“
  – „pass“
  – „observe“
  – „find“




## Informationsklassen

- hierarchische Informations- bzw. Geheimhaltungsklassen
- Informationsflüsse sind nur von unten nach oben erlaubt
- Relation bildet Totalordnung

1. streng geheim/top secret
2. geheim/secret
3. vertraulich/confidential
4. offen/unclassified

## Bell-LaPadula-Modell

- besteht aus einem Neun-Tupel $(S, O, M, C, cl, \leq, E, \delta, q_o)$
  - S, O: Menge von Subjekten und Objekten
  - M: Zugriffsmatrix S x O mit read, write
  - C: Menge der Sicherheitsklassen/Geheimhaltungsstufen
  - cl: Zuordnung zur Sicherheitsklasse $S \cup O \rightarrow C$
  - ≤: Ordnungsrelation, die zulässige Informationsflüsse beschreibt

    - E, $\delta$: mögliche Änderungsoperationen und deren Übergänge
    - $q_0$: Ausgangszustand
- Simple Security Role SSR (read down) / lesesicher: Zustand ist lesesicher, wenn für alle Subjekte und Objekte mit read gilt, dass cl(o) ≤ cl(s)
- *-Property (write up) / schreibsicher: Zustand ist schreibsicher, wenn für alle Subjekte und Objekte mit write gilt, dass cl(o) ≤ cl(s)
- Sicherer Zustand: Zustand ist sicher, wenn er lese- und schreibsicher ist
- Sicheres BLP-Modell: Modell heißt sicher, wenn Startzustand $q_0$ sicher ist und jeder erreichbare Zustand ebenfalls sicher ist

## Flusskontrolle

- An welcher Stelle und wie können Flüsse in Rechnern auftreten?
- offener Kanal: vom System für Transport von Informationen vorgesehen
    - expliziter Fluss: basieren auf explizitem Transportbefehl (z.B. strcpy)
    - impliziter Fluss: entstehen durch Bedingungen
- verdeckter Kanal: Informationen fließen, aber nicht vorgesehen → Laufzeiten, Fehlermeldungen, Seiteneffekte, Abstrahlung

### Umsetzung der Flusskontrolle

- Laufzeit:
    - Prozessor hat spezifische Register, die Sicherheitsstufen von Prozessen und Objekten prüfen
    - Prozess und Objekte haben feste Sicherheitsstufen
- Übersetzungszeit:
    - Compiler kann ähnliche Kontrollen während der Übersetzung durchführen
    - alle Pfade geprüft, aber ohne Kontext
    - erkannte Sicherheitsverletzungen führen zu Fehlermeldungen

# ▼ IT-Grundschutz

- Standard vom BSI

- im BSI-Gesetz werden Aufgaben des IT-Grundschutzes geregelt:
  - Gefahrenabwehr
  - Sammlung von Infos über Sicherheitsrisiken
- nationaler Standard zur Informationssicherheit
- Anwendung für Bundesbehörden verpflichtend
- nationale Zertifizierung möglich

## Standards des BSI

- BSI 200-1: Informationssicherheitmanagementsystem
- BSI 200-2: IT-Grundschutz Vorgehensweise → wie erstellt man Sicherheitskonzept?
- BSI 200-3: Risikoanalyse
- BSI 100-4: Notfallmanagement
- Grundschutz-Kompendium: Sammlung von Standard-Sicherheitsmaßnahemn zur Umsetzung eines Basisschutzniveaus (Grundschutz)

## Grundidee

- wiederkehrende Grundkomponenten in IT-Umgebung
- vergleichbare IT-Komponenten sind vergleichbaren Bedrohungen ausgesetzt
- Ziel: Standard-Sicherheitsmaßnahmen für Standard-Risiken → Maßnahmen bei Organisation, Personal, Infrastruktur und Technik
- Hilfestellung für Unternehmen

## Vorgehen

- Definitionen von Bausteinen, die Komponenten abbilden (z.B. Server, Router, Besprechungsraum…)
- Katalog mit Mindestsicherheitsanforderungen für jeden Baustein→ Abgleichen zwischen Anforderung und Umsetzung
- detaillierte Umsetzungshinweise mit empfohlenen Maßnahmen

- Problem bei standartisierter Vorgehensweise: Schutzbedarf hängt auch von Wichtigkeit der verarbeiteten Daten etc. ab → es können Schutzlücken entstehen und auch Overprotection
- erhöhte Schutzbedürfnisse müssen individuell behandelt werden

## IT-Grundschutz-Methode

1. Inititiierung des Sicherheitsprozess → Bereitstellung von Ressourcen, Entscheidung für eine Vorgehensweise, Konzeption und Planung des Sicherheitsprozesses
2. Erstellung der Leitlinie zur Informationssicherheit
3. Organisation des Sicherheitsprozesses
4. Erstellung einer Sicherheitskonzeption
5. Umsetzung der Sicherheitskonzeption
6. Aufrechterhaltung und Verbesserung

---

- zunächst muss Geltungsbereich definiert werden
  - Informationsverbund
  - Gesamtheit von infrastrukturellen, organisatorischen und technischen Komponenten
  - ein Informationsverbund kann aus gesamter IT oder einzelnen Bereichen bestehen
  - Informationsverbund muss beschrieben und abgegrenzt werden
  - eine Auflistung von Komponenten ist nicht ausreichend zur Abgrenzung des Informationsverbundes
- dann muss der Ist-Zustand analysiert werden
  - Listen von Komponenten des Informationsverbundes
  - Geschäftsprozesse, Anwendungen etc. auswerten
  - alles, was dazu gehört, muss betrachtet werden (Räume, Netzplan, Abhängigkeiten…)
  - Gruppierung von Dingen mit gleichem Typ, ähnlicher Konfiguration, ähnlicher Einbindung ins Netz, ähnlichen Bedingungen, ähnlichen

Anwendung und gleichem Schutzbedarf zur Komplexitätsreduktion spart redundante Arbeit

- so kann der Schutzbedarf festgestellt werden
  - Bestimmung der möglichen Schäden
  - High-level-Risikoanalyse (nicht tief im technischen Detail)
  - Beachtung der Sicherheitsziele Vertraulichkeit, Integrität und Verfügbarkeit
  - Ermittlung des jeweils höchsten plausiblen Schadens
  - Schutzbedarf wird in Kategorien eingeteilt
    - normal: Schadensauswirkungen begrenzt und überschaubar → Schutzbedarf typischer Komponenten, Standardsicherheitsmaßnahmen nach IT-Grundschutz ausreichend und angemessen
    - hoch: Schadensauswirkungen können beträchtlich sein → Standardsicherheitsmaßnahmen nach IT-Grundschutz reichen evtl. nicht, weitergehende Maßnahmen auf Basis einer ergänzenden Sicherheitsanalyse
    - sehr hoch: Schadensauswirkungen können existenziell bedrohliches Ausmaß entfalten → Standardsicherheitsmaßnahmen nach IT-Grundschutz reichen im Allgemeinen nicht, weitergehende Maßnahmen auf Basis einer ergänzenden Sicherheitsanalyse
  - Kategorien sind je nach Branche unterschiedlich
  - man kann Schadensszenarien betrachten, um Sicherheitsziele zu bewerten (finanzielle Auswirkungen, Gesetzesverstöße, Beeinträchtigung der Unversehrtheit etc.) → Kritikalitätsmatrix Schutzkategorie x Schadensszenario
- dann erfolgt Modellierung und ein Soll-Ist-Vergleich
- nach Risikoanalyse und Konsolidierung folgt ein erneuter Check und eine Realisierung der Maßnahmen
- wichtig: es braucht kontinuierliche Verbesserung

## Hilfestellung bei der Bewertung - Schadensszenarien

- Verstoß gegen Gesetze, Vorschriften, Verträge
- Beeinträchtigung des informationellen Selbstbestimmungsrechts (Datenschutverstöße)
- Beeinträchtigung der persönlichen Unversehrtheit (Gefahr für Leib und Leben)
- Beeinträchtigung der Aufgabenerfüllung
- negative Innen- und Außenwirkung
- finanzielle Auswirkungen

## Kritikalitätsmatrix

- Matrix mit Kriterien für den Schutzbedarf
- links stehen Schadensszenarien
- oben steht Schutzbedarf (normal, hoch, sehr hoch)
- in den Zellen stehen Kriterien

## Schutzbedarfsvererbung

- Maximum-Prinzip: wenn sich Schutzbedarf eines Objekts aus Abhängigkeit von anderen Objekten ergibt, dann richtet sich der Schutzbedarf nach höchstem Schutzbedarf der Objekte
- Kumulationsprinzip: wenn von einem Objekt viele andere Objekte abhängen, dann kann diesem Objekt ein höherer Schutzbedarf gegeben werden als der von den abhängigen Objekten
- Verteilungseffekt: wenn ein Objekt von einem anderen Objekt abhängt, kann ihm dennoch ein geringerer Schutzbedarf zugewiesen werden, wenn Objekt für die Erfüllung des Schutzbedarf des abhängigen Objekts nicht wesentlich ist

## Kommunikationsverbindungen

- gelten als kritisch, wenn es sich um Außenverbindungen handelt oder Informationen mit erhöhtem Schutzbedarf übertragen werden

## Modellierung im IT-Grundschutz

- Auswahl anwendbarer Prozessbausteine
- Zuordnung von Zielobjekten zu Systembausteinen
- ein oder mehrere Bausteine pro Zielobjekt
- Festlegung, wie man mit Zielobjekten umgeht, die nicht passen

## Struktur von Bausteinen

1. Beschreibung
2. spezielle Gefährdungen und elementare Gefährdungen
3. Anforderungen (Basis-, Standard- und Anforderungen für erhöhten Schutzbedarf)
4. Umsetzungshinweise

→ Prozessbausteine sind idR auf den gesamten Informationsverbund anzuwenden

## Umgang mit Komponenten, die nicht auf Bausteine passen

- Möglichkeit 1: Vormerken für Behandlung in ergänzender Risikoanalyse
- Möglichkeit 2: Gestaltung eines “benutzerdefinierten Bausteins” → Zuordnung zu Gefährdungen und Maßnahmen (besonders bei mehrfacher Anwendung des Bausteins im Informationsverbund empfohlen)

## IT-Grundschutz-Check

- Ermittlung von Sicherheitsanforderungen aus Buasteinen
- Streichen von Anforderungen, die nicht anwendbar sind

### Prioritätsklassen von Grundschutz-Anforderungen

- Basisanforderungen: vorrangig, Risikoübernahme nicht zulässig
- Standardanforderungen: müssen umgesetzt werden
- Anforderungen für erhöhten Schutzbedarf: zusätzlich, können bei erhöhtem Schutzbedarf geprüft werden

### Ergebnis der Umsetzungsprüfung im Grundschutz-Check

- entbehrlich: Umsetzung nicht erforderlich, da bereits andere Maßnahmen gegen Gefährdung wirken oder da Anforderung irrelevant → Gründe dokumentieren
- ja: alle Aspekte sind vollständig, wirksam und angemessen umgesetzt → Umsetzung dokumentieren
- nein: größtenteils noch nichts umgesetzt
- teilweise: einige Aspekte umgesetzt, andere nicht → dokumentieren, was fehlt

⇒ am Ende kann man Grundschutz-Anforderungen und umgesetzte Maßnahmen vergleichen und findet so defizitäre Maßnahmen

## Risikoanalyse

- durchzuführen für…
  - … Objekte mit hohem oder sehr hohem Schutzbedarf
  - … Objekte, die durch Bausteine nicht (geeignet) abgebildet werden
  - … Objekte in besonderen Einsatzszenarien

## Umsetzung IT-Grundschutz

- Konsolidierung von Maßnahmen aus Grundschutz-Check und Risikoanalyse
- Festlegung von Verantwortlichkeiten, Prioritäten, Umsetzungszeiträumen

## Alternative Absicherungen

- Basis-Absicherung: zuerst Basisanforderungen umsetzen, dann Standardanforderungen
- Kern-Absicherung: Umsetzung zuerst für kritische Bereiche, danach für alles andere

## Zertifizierung

- Nachweis, dass Informationssicherheit systematisch verfolgt wird
- Nachweis, dass mindestens standardisiertes Schutzniveau erreicht ist

- Vertrauensbasis
- Verpflichtung, Sicherheitsprozess aufrecht zu erhalten

### Verfahren

- Beantragung beim BSI
- Vorbereitung 12-36 Monate, Audit 15-35 Tage und Zertifizierungsgebühr
- Gültigkeit drei Jahre, jährliche Audits zur Überwachung

## IT-Grundschutz vs. ISO 27001

- aussagekräftiger (Prüfung tatsächlicher struktureller, personeller, organisatorischer Maßnahmen) vs. überwiegend Konzepte
- konkrete Hinweise vs. allgemeine Empfehlungen
- maßnahmen- vs. prozessprientiert
- Risikoanalyse nur für spezielle Dinge vs. Mehraufwand durch komplette Risikoanalyse
- drei Jahre gültig mit jährlichem Überwachungsaudit vs. drei Jahre gültig mit jährlichem Continuing Assessment Visit

# ▼ Datensicherung

→ es soll verhindert werden, dass Daten verloren gehen: Schutzziele Integrität und Verfügbarkeit

## Ursachen/Bedrohungen

- Benutzerfehler
- Fehler des Betriebspersonals
- Hardwarefehler
- Softwarefehler
- Vandalismus durch Hacker oder Viren
- Diebstahl
- Naturkatastrophen
- andere Katastrophen

## Mögliche Lösungen

- Transparente Redundanz
  - Fehler automatisch ausgleichen
  - zeitnah und meist ohne Verlust
  - hilft nur gegen Fehler in unteren Schichten (Festplattenfehler, Hardwarefehler, Systemausfall)
  - Fehleingaben werden umgesetzt
- Backups
  - Fehler werden später behoben
  - meist nur diskrete Sicherung
  - kann gegen Fehler in höheren Schichten (Viren, Fehleingaben) helfen

## RAID - Transparente Redundanz

- mehrere Festplatten zusammenschalten als Teil des OS oder in Hardware
- Redundancy Array of Independet Discs
- man speichert Daten redundant
- Level 0: doppelte Kapazität, aber kein Ausfallschutz
- Level 1: einfache Kapazität, schützt aber gegen einen Festplattenfehler
- Level 5: n-1 Kapazität, schützt gegen einen Festplattenfehler

### Failover - Transparente Redundanz

- komplettes System noch ein mal vorhalten und im Fehlerfall dort weitermachen
- beide Systeme synchron: muss von Software unterstützt werden
- Hot-Standby: Zweitsystem läuft immer und pflegt Datenänderungen vom primären System nach
- Cold-Standby: Zweitsystem wird erst bei Fehler hochgefahren

### Probleme bei transparenter Redundanz

- kein Schutz vor Fehlern auf höheren Protokollschichten (Unachtsamkeit, Bedienfehler, Softwarefehler)
- meist Single Point of Failure (RAID-Controller oder Failover-Komponente)
- neue Fehlerquellen durch komplexen Aufbau und zusätzliche Komponenten
- kein Schutz vor Diebstahl oder Katastrophen ohne räumliche Trennung

## Sicherungskopien/Backups

- System kann wieder in älteren Zustand zurückversetzt werden
- ältere Sicherungskopien können ältere Fehler rückgängig machen
- Zeitpunkt diskret: seltene Sicherungen führen zu ungesicherten Änderungen
- aber bei sehr häufigen Sicherungskopien viele Daten
- man braucht Sicherungsstrategie, die an Bedürfnisse angepasst ist

## Arten von Sicherungsläufen

- Anfangssicherung (day zero backup)
  - Kopie des initialen Systems bei Inbetriebnahme
  - alle Dateien und Programme
- Komplettsicherung (full backup)
  - Kopie jeder einzelnen Datei
  - wie bei Anfangssicherung, aber in regelmäßigen Abständen oder bei Installation neuer Softwaresysteme
- Differenzielle Sicherung (differential backup)
  - eine Kopie jeder Datei, die seit Komplettsicherung modifiziert wurde oder hinzu kam
- Inkrementelle Sicherung (incremental backup)
  - eien Kopie jeder Datei, die seit letzter Sicherung modifiziert wurde oder hinzukam

⇒ am besten Kombination verwenden!

### Tandem Strategie

- zwei verzahnte Folgen
- Schutz vor Fehlern des Backup-Mediums

### Sicherungsmedien

- alles, worauf man aktuell Daten speichern kann
- Bänder/Bandkassetten: klassisches Medium
- Festplatten
- optische Medien (CD,DVD, Blu-Ray)
- Onlinespeicher in der Cloud

### Sicherung der Sicherungskopien

- fest mit dem System verbundene Medien helfen nur begrenzt gegen einige Bedrohungen
- man sollte Duplikate von Komplettsicherungen anlegen und an geographisch anderen Orten lagern
- man sollte alte Bänder und Typen auf neue Medien kopieren, solange sie noch lesbar sind

## ▼ Informationssicherheits-Managementsysteme

### Managementsysteme Definiton

- Gesamtheit von Richtlinien, Prozessen, Verfahren und zugehörigen Ressourcen und Fähigkeiten, die Ziel verfolgen, Managementaufgaben eines bestimmten Fachgebietes effektiv auszuführen

### Fragen des Information Security Management

- Sicherheitsanforderungen?
- Wie erfüllt man die Sicherheitsanforderungen?
- Sicherheit aufrechterhalten?
- Restrisiken?

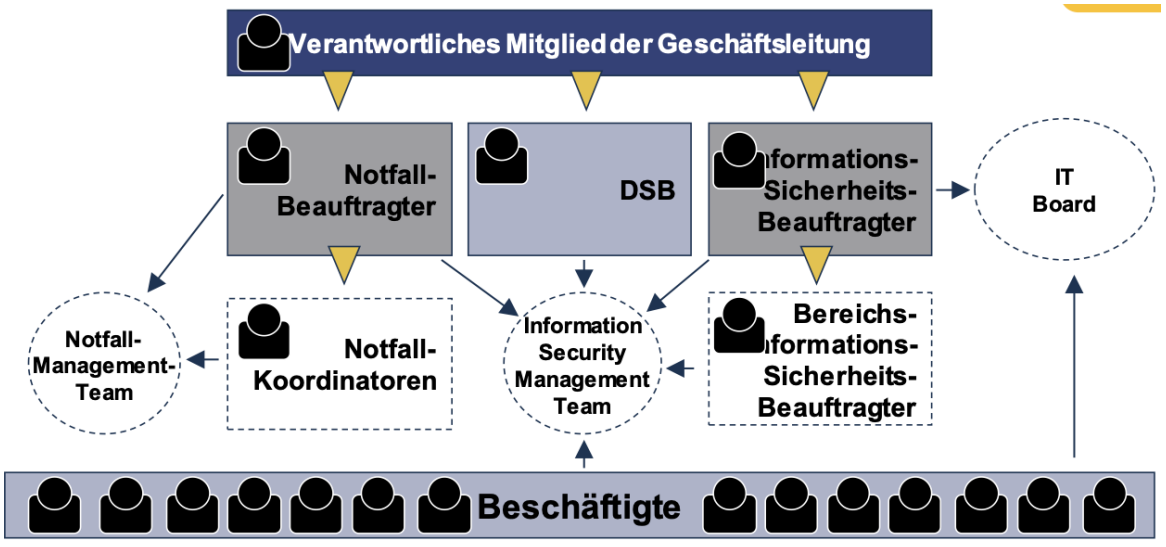
# Information Security Management in Organisation

- Governanve und Copmpliance stellen Anforderungen an ISM
- ISM bekommt Entscheidungen vom Top Management und erstattet an dieses Berichte
- Restrisiken werden vom Risikomanagement bearbeitet

# Management Commitment

- kritisch für das ISMS
- autorisiert Informationssicherheitsbeauftragten
- betont Stellenwert der Informationssicherheit für Unternehmen
- Grundlage der Durchsetzung von Regelungen
- zeigt, dass Regeln für alle gelten (management by example)

# Aufbau




# Rolle des Informationssicherheitsbeauftragten

- steuert Informationssicherheitsprozess
- entwirft und pflegt Informationssicherheit-Leitlinie
- veranlasst Erstellung eines Sicherheitskonzept und hält es aktuell
- steuert und überprüft Umsetzung von Sicherheitsmaßnahmen

- berichtet an Leitungsebene
- wird in Projekte mit Auswirkungen auf Sicherheit eingebunden
- steuert Behandlung von Sicherheitsvorfällen
- initiiert laufende Maßnahmen für Sensibilisierung der Mitarbeiter

### Probleme bei der organisatorischen Einbindung

- in IT: Interessenkonflikte, Risiken verschweigen
- außerhalb IT: Informationsdefizite
- IT-Leiter: fehlende Zeit, undokumentierte Interessenkonflikte zwischen Sicherheit und betrieblichen Anforderungen
- niedrige Hierarchiestufe: Einwände werden ignoriert oder überstimmt
- hohe Hierarchiestufe: fehlende Zeit, gefilterte und geschönte Infos

## PDCA-Zyklus

- Plan: Entscheiden, was man tut → Kontext der Organisation, Führung, Planung
- Do: Plan ausführen → Unterstützung und Ausführung
- Check: Resultate ansehen → Auswertung
- Act: Verbesserung

⇒ PCDA erfordert laugende Verbesserung eines Managementsystems

### Reifegradmodelle

- der erreichte Stand bei PDCA kann mit Reifegradmodellen verfolgt werden
- Beispiele CoBit oder SSECMM
- 0: Nicht-existent → keine Prozesse in Planung
- 1: initial → ad-hoc Prozesse
- 2: repeatable → Prozesse folgen einem simplen Muster
- 3: defined → Prozesse sind dokumentiert
- 4: managed → Prozesse werden aufgezeichnet und gemessen
- 5: optimised → reguläre Anwendung der Korrekturmaßnahmen

## Relevante Standars

- verschiedene relevante Standards
- international ISO 27001 → ISMS Requirements, und folgende
  - Allgemeine Aussagen über Information Security Management Systeme
  - Zertifizierung durch akkreditierte Zertifizierungsunternehmen
  - sehr generischer Standard (im Umsetzung große Freiheitsgrade) → kein Anleitungscharakter
  - erfordert immer eine Risikoanalyse
  - 27002: Umsetzungsempfehlungen
    - Empfehlungen auf Managementebene, kaum konkrete technische Hinweise
    - Zertifizierung nach ISO ist grundsätzlich nicht möglich

  1. Scope: Festlegung des Betrachtungsgegenstandes und Gültigkeitsbereiches
  2. Asset: Aufbau eines Assetmanagements zur Inventarisierung aller den ISMS-Bereich betreffenden Assets
  3. Risiko: Definition eines Verfahrens zur Risikoabschätzung und Festlegung der Kriterien für Risikoakzeptanz
  4. Controls: Sicherheitsmaßnahmen
  5. SOA: Statement of Applicability (SOA), Nachweis der Erfüllung der Schutzziele
- national BSI 200-1 → ISMS und folgende

## Risikomanagement → Wieso Risikoanalyse?

- Risikosituationen transparenter machen
- Maßnahmen planen
- Maßnahmen priorisieren
- angestrebtes Schutzniveau durchgängig erreichen
- für Zertifizierung benötigt

### Ausrichtung an Standards

- ISO 27005: internationaler Standard mit sehr allgemeiner Rahmenvorgabe, keiner Festlegung der Risikobewertungsmethode → sehr viel Ausgestaltungsspielraum
- BSI 200-3: Nationaler Standard mit konkreter Handlungsvorgabe und zweidimensionaler Risikobewertung nach Eintrittshäufigkeit und Schadenshöhe

## Risk Identification/Gefährdungsübersicht

- Ermittlung der Bedrohungen
  - Erfahrung
  - Expertenwissen
  - Bedrohungskataloge
- Vorhandene Maßnahmen
  - sollten bekannt oder ermittelbar sein
- Schwachstellen
  - vollständige Erhebung nicht möglich
  - man kennt nicht alle Schwachstellen
  - wenn Schwachstellen bekannt, dann Berücksichtigung

## Risk Analysis/Risikobewertung

- Kombination aus Auswirkungen und Eintrittswahrscheinlichkeit
- Auswirkungen und Wahrscheinlichkeit qualitativ oder quantitativ bewerten

## Risk Evaluation

- Entscheidung, ob Handlungsbedarf besteht
- Grundlage: Risikobewertung, Relevanz der Sicherheitsziele (CIA), Relevanz der Geschäftsprozesse

## Risk Treatment/Behandlung von Risiken

- Risk Modification, Risikoreduktion durch weitere Maßnahmen

- Risk Avoidance, Risikovermeidung durch Umstrukturierung
- Risk Retention, Risikoübernahme
- Risk Sharing, Risikotransfer

### Best Practice

- lieber wenige und relevante Risiken ausführlich behandeln, als viele Risiken oberflächlich
- Gedankengänge und Abwägungen dokumentieren (Text), nicht nur Zahlen
- Vorsicht bei Zahlen: Gefahr der Vortäuschung von Genauigkeit
- komplexe Bewertungsverfahren bringen selten Erkenntnisgewinn und binden viel Kapazität

## ▼ Verschlüsselung

### Grundprinzip

- Klartext M → Chiffre E(M) → Klartext M = D(E(M))
- man verschlüsselt Nachricht M als Chiffre mit Funktion E und diese wird dann vom Empfänger mit zu E inverser Funktion D entschlüsselt
- meistens handelt es sich bei E aber um eine Relation (nicht immer eindeutig)
- es geht um Vertraulichkeit von Nachrichten → klassisches Einsatzgebiet
- auch für gespeicherte (nicht nur übermittelte) Infos
- Prüfung der Authentizität von Personen oder Daten (digitale Unterschrift)
- Prüfung der Unverfälschtheit einer übertragenen Nachricht sowie der Authentizität von Daten

### Einfache Verschlüsselungsverfahren

- **Caesarchiffre** → jeden Buchstaben durch dritten Nachfolger ersetzen
- **Shift-Chiffre** → Verschieben um k Buchstaben, hat Parameter (k als geheimen Schlüssel)
- Monoalphabetische Substitution: jede beliebige Umsortierung des Alphabets, 26! Möglichkeiten

- nach wie vor unsicher: Häufigkeitsanalysen (auch andere Anwendungen wie Buchdruck, Morsecode (häufige Buchstaben, kurze Zeichen), Datenkompression)
- auch Paare oder Tripel für typische Häufigkeiten
- darf für sicheres Verfahren nicht gehen!

- **Vigenère-Verfahren**: ein Zeichen nicht immer auf dasselbe abbilden, der Schlüssel ist ein Wort → Schlüssel und Klartext werden zeichenweise addiert (Entschlüsselung dann per Subtraktion)
  - traditionell keine Leerzeichen, Groß- und Kleinschreibung, geht aber überall

**Beispiel**

Nachricht "IDEFIX", Schlüssel "ASTERIX"

$$\begin{array}{rr} & \texttt{IDEFIX} \\ + & \texttt{ASTERI} \\ \hline & \texttt{IVXJZF} \end{array} \quad A = 0, B = 1, \ldots$$

Beachte I→I und I→Z

- **Polyalphabetische Verfahren (verallgemeinertes Vigenère)**: beliebige monoalphabetische Chiffre pro Stelle → Schlüssel besteht dann aus verschiedenen Alphabeten statt aus einem Wort
  - Angriff: wenn man Länge des Schlüssels kennt, dann kennt man Verfahren für 1, 1+l, 1+2l → Häufigkeitsanalyse
  - gerade bei mehreren Texten beginnen sie immer mit demselben Schlüssel → man kann daher bei unterschiedlichen Stellen anfangen
  - Kasiskitest: man untersucht Wiederholungen von Symbolfolgen und nimmt an, dass dort Schlüssel und Klartext übereinstimmen → man misst dann Länge zwischen Folgen und die Schlüssellänge ist der ggT dieser Abstände
- **Vernam-Chiffre (Spezialfall Vigenère)**: Schlüssel so lang wie der Klartext, z.B. Teil eines Buchs

    - auch hier ist Häufigkeitsanalyse bei bekannter Wortlänge noch möglich
    - One Time Pads bieten besseren Schutz, aber können nur ein mal verwendet werden → meist unpraktikabel
- **Permutationsverfahren**: Buchstaben werden nicht ersetzt, sondern nur untereinander vertauscht → jeweils n Buchstaben zusammenfassen und nach Abbildung permutieren, Entschlüsselung durch Inverse
    - man braucht eine Regel für den letzten Block (so lassen, mit Leerzeichen auffüllen…)

## Anforderungen

- Blocklänge: Blocklänge groß genug, um Häufigkeitsanalysen zu vermeiden → 128-bit-Blöcke
- Schlüssellänge: Vollständiges Ausprobieren aller Schlüssel sollte unmöglich sein
- Konfusion: verschlüsselter Text sollte in möglichst komplexer Weise von Klartext und Schlüssel abhängen → keine statistischen Zusammenhänge sollen existieren
- Diffusion: jedes Bit des Klartextes und jedes Bit des Schlüssels sollte jedes Bit des verschlüsselten Textes beeinflussen

## Angriffsarten

- Ciphertext-only-attack: Angreifer kennt nur verschlüsselten Text (Abfangen und Mithören von Nachrichten)
- Known-plaintext-attack: Angreifer kennt sowohl Chiffretext als auch Klartext → z.B Kenntnis von Teilen der Nachricht, wie E-Mail-Header
- Chosen-plaintext-attack: Angreifer kann Klartext selbst wählen und verschlüsseln lassen
- Chosen-ciphertext attack: Angreifer kann von ihm kontrollieren Chiffretext entschlüsseln lassen
- theoretisch geht auch immer Bruteforce

## Maximen

- Angreifer besser über- als unterschätzen

- nur ein Kryptoanalytiker kann Sicherheit beurteilen
- Kerckhoffs Prinzip: Es ist davon auszugehen, dass Angreifer Algorithmus kennt, aber den Schlüssel nicht → je mehr Leute probieren, zu knacken, desto besser
- zusätzliche Komplikationen machen Verfahren nicht unbedingt sicherer (kein Security by Obscurity)
- menschliche Fehler müssen berücksichtigt werden

# Symmetrische Verschlüsselung

- Verschlüsselungsalgorithmus E besteht idR aus Menge von Verschlüsselungsfunktionen, aus denen durch Vorgabe eines Parameters (Schlüssel) eine bestimmte gewählt wird
- Wahl des Schlüssels sollte erheblichen Einfluss auf Entschlüsselungsfunktion D besitzen
- beide Teilnehmer haben einen gemeinsamen Schlüssel
- nur der Schlüssel muss geheim gehalten werden, E und D sind bekannt
- Problem: Wie wird der Schlüssel bekannt?

# AES - Advanced Encryption Standard

- Nachfolger von DES
- Anforderungen:
  - Blockverschlüsselung: 128-bit-Blöcke
  - Schlüssellänge: 128/192/256 bit
  - effiziente Implementierbarkeit
  - Dokumentation, C- und Javacode
  - weltweit freie Verfügbarkeit
- Gewinner wurde Algorithmus Rijndael

## AES (Rijndael)

- Blocklänge: 128 Bit
- Schlüssellänge 128/192/256 Bit

- Runden:
  - 10, wenn Block und Schlüssel 128 Bit
  - 12, wenn Block oder Schlüssel 192 Bit
  - 14, wenn Block oder Schlüssel 256 Bit
- jede Runde besteht aus vier Schritten:
  1. (128-Bit als 4x4 Matrix von Bytes schreiben)
  2. Byte-Substitution: jedes Byte nach gewissen Schema ersetzen (Lookup-Tabelle, öffentlich bekannt)
  3. Permutation: Zeilen zyklisch um 0,1,2 bzw. 3 Byte nach links verschieben
  4. Mix-Column: jede Spalte der Matrix mit einer festen Matrix C multiplizieren (Einträge aus der Spalte mischen)
  5. Add-Round-Key: bit-weises xor mit dem Rundenschlüssel, abgeleitet aus geheimen Schlüssel K → erstes Mal wird Schlüssel genutzt!!!

## Mehr Daten, als in einen Block passen: Electronic Codebook

- jeder Block im Klartext in Block-Chiffre transformiert
- gleiche Klartextblöcke auf gleiche Schlüsseltextblöcke abgebildet
- bei längeren Texten vereinfacht das einen Angriff → ECB nur für kurze Nachrichten verwenden
- → daher weitere Verfahren wie Cipher Block Chaining, die mit Verkettung arbeiten, wo jeder Block vom nächsten abhängt → XOR mit vorherigem Output
  - der erste Block wird mit Initialisierungsvektor verschlüsselt (damit nicht immer auf dieselbe Art und Weise vorgegangen wird)

## Stromverschlüsselung (Stream Cipher)

- so viele Daten verschlüsseln, wie gerade vorliegen
- angelehnt an One-Time-Pad, verschlüsselt kleine Einheiten (Bits oder Byte)
- Schlüsselstrom
  - basiert auf Schlüssel, Nonce oder IV

    - möglichst zufällige und lange Periode
    - sollte nicht zwei mal verwendet werden
    - Fehler bei Generierung zerstören Eigenschaften
    - jedes Klartextzeichen wird sofort chiffriert → besonders geeignet für Echtzeitübertragungen

## Problem des Schlüsseltausches

- symmetrische Verfahren sind sicher und schnell
- sie benötigen für jede Kommunikationsverbindung einen individuellen Schlüssel
- Kein Problem: Bei n Knoten müssen bis zu n(n-1)/n Schlüssel ausgetauscht werden
- symmetrische Verfahren setzen voraus, dass der geheime Schlüssel bereits sicher ausgetauscht wurde
- alternativ: zwei Schlüssel, einer davon darf öffentlich sein

## Asymmetrische Verschlüsselung

- jeder Teilnehmer hat zwei verschiedene Schlüssel:
    - öffentlicher Schlüssel X → hat jeder
    - privater Schlüssel Y → hat nur die Person selbst
- Klartext $M$, Chiffre $E_X(M)$
- notwendige Eigenschaft: $D_Y(E_X(M)) = M$
- wünschenswerte Eigenschaft: $E_X(D_Y(M)) = M$
- klassische Verfahren:
    - RSA
    - ElGamal
    - DSA
    - Elliptic Curve Cryptography

## Das RSA-Verfahren (Rivest, Shamir, Adleman)

## Zahlentheoretische Vorbemerkungen

Sei $\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$ und $a, b \in \mathbb{Z}_n$, dann sind Summe $\oplus$ und Produkt $\otimes$ wie folgt definiert:

$$a \oplus b = (a + b) \bmod n$$
$$a \otimes b = (a \cdot b) \bmod n$$

- bezüglich Addition bildet $(Z_N, \oplus)$ eine kommutative Gruppe, zusammen mit der Multiplikation bildet $(Z_n, \oplus, \otimes)$ einen Ring (sog. Restklassenring)
- wenn $a \otimes b = 1$, dann heißt b das multiplikative Inverse zu a, geschrieben $b = a^{-1}$
- $a^{-1}$ gibt es genau dann, wenn a teilerfremd zu n ist, d.h. der GGT von a und n ist 1
- entfernt man aus $Z_N$ alle Elemente, die mit n einen gemeinsamen Teiler haben, erhält man $Z_n^* = \{a \in Z_n : ggT(a, n) = 1\}$
- $(Z_N^*, \otimes)$ ist eine multiplikative Gruppe → Zahl der Elemente in dieser Gruppe heißt $\varphi(n)$ (Eulersche Phi-Funktion)
- kennt man alle Primteiler von n, gilt: $\varphi(n) = \prod(p_i - 1)p_i^{e_i - 1}$

## Spezialfälle

- ist p eine Primzahl, so gilt $\varphi(p) = p - 1$
- ist $n = pq$ ein Produkt von zwei verschiedenen Primzahlen, so gilt $\varphi(n) = (p-1)(q-1)$

## RSA-Schlüsselerzeugung

- wähle zwei zufällig große Primzahlen (größer als 1024 Bit) p und q
- setze $n = pq$, also gilt $\varphi(n) = (p-1)(q-1)$
- wähle eine Zahl $e$, die teilerfremd zu $\varphi(n)$ ist → keinen gemeinsamen Teiler außer 1
- berechne mit erweitertem euklidischen Algorithmus $d := e^{-1} mod\ \varphi(n)$ → es muss für d also gelten, dass $ed\ mod\ \varphi(n) = 1$

⇒ öffentlicher Schlüssel (n, e)

⇒ privater Schlüssel (n, d)

- Verschlüsselung $C = E_{n,e}(M) = M^e mod\, n$
- Entschlüsselung $M' = D_{n,d}(C) := C^d mod\, n$
- Klartexte, Chiffre sind Zahlen aus $Z_n$, also $M, C < n$
- Text ist eine ASCII-Zahl mit Basis 256
- Blocklänge ist k Bit, also muss man p und q groß genug wählen, sodass $2^k < n$

## RSA in der Praxis

- Sicherheit wird in der Größe von n bemessen, aktuell 2048 oder 4096 Bit → Empfehlung mindestens 3072 Bit
- Primzahlen p und q sollten etwa gleich viele Bit haben
- man erhält große Primzahlen, indem man zufällige Zahlen erzeugt und testet, ob sie prim sind
- e = 65537 ($2^{16}$ + 1) → notfalls kann e auch 3 sein, dann schneller
- Quantencomputer könnten allerdings in mittlerer Zukunft Zahlen faktorisieren und damit RSA brechen (AES ist zukunftssicherer)

Wenn ich nur für mich was verschlüssele, dann ist AES sinnvoller, weil ich ja gar keinen Schlüssel austauschen muss. Außerdem ist AES performanter!

## Diffie-Hellman

- es reicht, wenn Alice und Bob ein paar Nachrichten austauschen und am Ende ein gemeinsames Geheimnis haben → dieses dann Schlüssel für symmetrische Verfahren (erstes asymmetrisches Kryptoverfahren!!!)
- ermöglicht, dass zwei Kommunikationspartner über öffentliche Leitung einen gemeinsamen, geheimen Schlüssel vereinbaren können, den nur sie selbst kennen
- man hat öffentlichen Schlüssel p=13 und g=2 und jeder hat einen privaten Schlüssel (a = 8, b = 5)
- geheime Werte werden mit Einwegfunktion berechnet → $A = g^a\ mod\ p$ und $B = g^b\ mod\ p$

- Partner schicken sich jeweils ihren errechneten geheimen Werte an das Gegenüber
- jetzt kann jeder den gemeinsamen Schlüssel berechnen: $K = B^a \ mod \ p$ bzw. $K = A^b \ mod \ p$

## Signaturen

- Teil asymmetrischer Kryptographie
- einfach mit RSA umsetzbar
- auch Verfahren angelehnt an Diffie-Hellmann (DSA/ECDSA)

### Signaturen mit RSA

- m ist Nachricht, (n,e) bzw. (n,d) sind öffentlicher und privater Schlüssel
- Signatur $s = m^d mod \ n$ → Alice übermittelt $(m, s)$
- Bob erhält $(m', s')$, eventuell von Eve manipuliert, und prüft ob $m' = s'^e mod \ n$
- funktioniert, weil $(m^d)^e \equiv m^{ed} \equiv m \ mod \ n$

### Hashing

- Funktion $h : \{0,1\}^* \rightarrow \{0,1\}^n$ für einen Sicherheitsparameter n
- schwierig, Urbild zu berechnen und zwei Argumente mit gleichem Funktionswert zu finden

### Hash-then-sign

- signieren $h(m)$ statt $m$
- Nachricht nicht mehr durch RSA-Modulus begrenzt
- man will schnellen Hash

# ▼ Public Key Infrastrukturen und die elektronische Signatur

## Was ist eine Unterschrift?

- Abschlussfunktion: bestätigt, dass das vorliegende Dokument einen fertigen Stand darstellt
- Perpetuierungsfunktion: hält abgegebene Willenserklärung fest
- Identitätsfunktion: bestätigt Identität von Unterzeichner
- Echtheitsfunktion: bestätigt, dass das Dokument dem Unterzeichner vorgelegen hat
- Verifikationsfunktion: ermöglicht Prüfung der Echtheit
- Beweisfunktion: ermöglicht Nutzung eines Dokumentes vor Gericht
- Warnfunktion: bestätigt, dass Unterzeichner bewusst war, mit Unterschrift eine Willenserklärung abzugeben

## Nachrichtenaustausch

- Wie kann man sichergehen, dass eine Nachricht sicher übertragen wird?
- Mit Einführen von Schlüsseln gibt es eine Verlagerung vom Problem des Austauschs der Nachricht auf den sicheren Austausch von Schlüsseln

## Asymmetrische Verschlüsselung

- mathematische Verfahren mit folgenden Eigenschaften:
  - Nutzung eines Schlüsselpaares
  - ein mit einem Schlüssel kryptierter Text kann nur mit dem anderen Schlüssel wieder lesbar gemacht werden
  - es ist nicht praktikabel möglich, aus einem Schlüssel den anderen zu ermitteln
- kritisch für Sicherheit ist die Geheimhaltung privater Schlüssel

## Signieren

- Alice will Nachricht an Bob schicken und signiert sie, indem sie die Nachricht mit ihrem privaten Schlüssel verschlüsselt
- Bob (und Eve/Angreifer) können sie mit öffentlichen Schlüsse von Alicel entschlüsseln und so sehen, dass sie wirklich von Alice kommt ist

## Verschlüsseln

- Bob will Nachricht an Alice schicken und verschlüsselt sie mit dem öffentlichen Schlüssel von Alice → nur Alice kann sie entschlüsseln

## Schlüsselmanagement

- vor jeder gesicherten Kommunikation müssen initial einmal die öffentlichen Schlüssel ausgetauscht werden
- bei n Teilnehmern gibt es n * (n-1) / 2 Austauschvorgänge → quadratisches Wachstum, jeder Teilnehmer muss (n-1) Schlüssel vorhalten

⇒ gemeinsames “Telefonbuch” bzw. Trustcenter:

- jeder hinterlegt seinen Schlüssel im Trustcenter
- wer kommunizieren will, schlägt Schlüssel nach und holt ihn sich
- Trustcenter muss Identität der Schlüsselinhaber prüfen
- Trustcenter muss seine Auskünfte vor Veränderung schützen
- generiert ein eigenes Schlüsselpaar
- öffentlicher Schlüssel des Trustcenters wird sicher an alle Teilnehmer verteilt
- Trustcenter prüft Identität aller Teilnehmer und signiert jeweils Datensatz mit dem öffentlichen Schlüssel und der Identität seine Inhabers

## Sperren

- wenn Schlüssel kompromittiert werden, werden sie einer Sperrliste hinzugefügt

### Zertifikatsprüfung offline mit Sperrliste

- die Sperrliste wird vom Trustcenter signiert
- Anwender laden sie herunter und ziehen sie bei Signaturprüfung heran
- Nachteil: kann sehr lang werden, außerdem vergeht Zeit zwischen Sperrung und Verteilung der neuen Liste

### Zertifikatsprüfung online mit OCSP-Responder

- bei Signaturprüfung wird OCSP-Dienst (Online Certificate Status Protocol) des Trustcenters online nach Status des Zertifikats gefragt

- mögliche Antworten:
  - gültig, gesperrt, unbekannt
- Antwort wird vom OCSP-Responder signiert → Server, der Auskunft über den Status von Zertifikaten geben kann

## Gültigkeitsprüfung

- Certificate Revocation Lists (CRL, Sperrlisten)
  - signierte Liste mit allen gesperrten Zertifikaten einer Certification Authority (CA)
  - kann zyklisch heruntergeladen und zur dezentralen Prüfung verwendet werden (Aktualitätsproblem)
  - Delta-CRLs mit Zertifikaten, die seit der letzten vollen CRL gesperrt wurden
- Online Certificate Status Control, OCSP
  - signierte Online-Gültigkeitsauskunft zu einem spezifischen Zertifikat
  - Auskunft “gültig”, “gesperrt” oder “unbekannt”
  - mit Übermittlung des Zertifikats (”abrufbar”) oder ohne (”nachprüfbar”)
  - Lastproblem

## Elektronische Signaturen

- Daten, die zu einem elektronischen Dokument hinzugefügt werden
- werden mit kryptographischen Verfahren erstellt
- machen nachträgliche Dokumentveränderungen erkennbar
- können nur vom Inhaber eines geheimen Signaturschlüssels erstellt werden
- ermöglichen es, über Zertifikat nachzuvollziehen, wer Signatur erstellt hat
- leisten keine Verschlüsselung

## Hash-Funktionen

- können verwendet werden, um Unverfälschtheit einer Nachricht zu gewährleisten

- eine Hash-Funktion H bildet eine Nachricht M auf einen Hashwert H(M) konstanter Länge ab → eine Art “Fingerabdruck” der Nachricht

### Eigenschaften kryptographischer Hash-Funktionen

- H ist praktisch nicht invertierbar, d.h. bei gegebenen Hash-Wert H(M) kann ; nicht ermittelt werden → preimage resistance
- H(M) hängt von jedem Bit in M ab, jede Modifikation von M muss zu anderem Hash-Wert führen
- es ist praktisch nicht möglich, zu einer gegebenen Nachricht M eine Nachricht M’ (Urbild) zu finden, für die gilt H(M) = H(M’) → schwache Kollisionsresistenz, 2nd preimage resistance
- ist es zusätzlich praktisch nicht möglich, ein Nachrichtenpaar M und M’ zu finden, für das gilt H(M) = H(M’) so heißt die Hash-Funktion starke Kollisionsresistenz → collision resistance

→ da der Wertebereich deutlich kleiner als der Definitionsbereich der Hashfunktion ist, wird es idR immer Kollisionen geben - sie zu finden sollte aber nicht leichter als Brute Force sein

### Signatur und Verifikation mit Hashverfahren

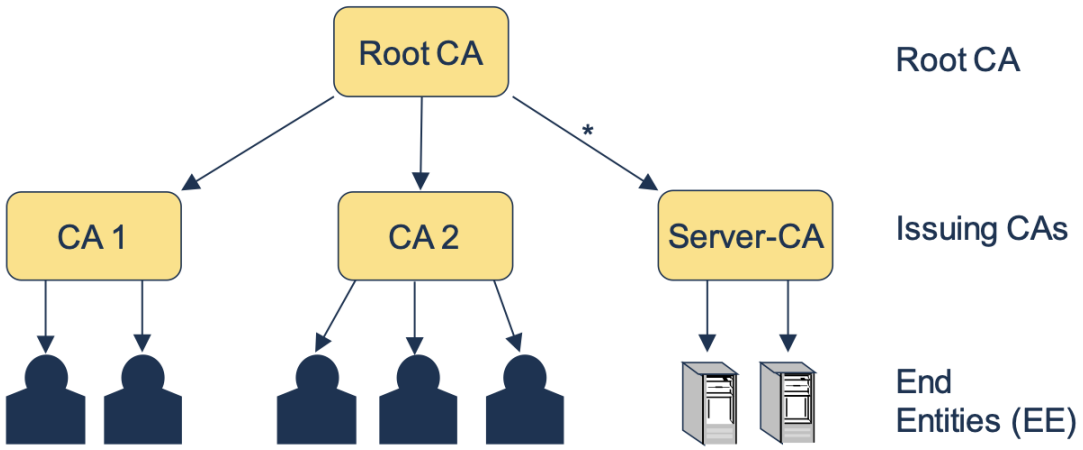
- Nachricht M sowie mit privatem Schlüssel von Alice verschlüsselter Hashwert der Nachricht werden von Alice an Bob verschickt
- Bob entschlüsselt den von Alice erhaltenen Hashwert mit ihrem öffentlichen Schlüssel, hasht ebenfalls die Nachricht und vergleicht die Hashwerte

## Signaturzertifikate

- Zertifikat ist ein überprüfbares, öffentliches Dokument
- enthält
  - Informationen über den Besitzer
  - öffentlichen kryptographischen Schlüssel
  - Verwaltungsinformationen
- ist von vertrauenswürdigen Institution (Zertifizierungsinstanz) unterschrieben/signiert

# CA-Hierarchien

- bestehen aus Root CA (oberste Zertifizierungsstelle), Issuing CAs und End Entities (EE)




## Cross-Zertifizierung

- zwei CA-Hierarchien
- Zertifizierung von Root 1 zu Root 2 und umgekehrt
- durch Cross-Zertifizierung können Zertifikate aus befreundeter Organisation auf eigene Root-CA zurückgeführt werden

# Trust Service Provider Lists (TSL)

- signierte Liste von Zertifizierungstellen und Zertifizierungsdiensten
  - enthält Angaben zum Anbieter und zum Dienst
  - enthält jeweils eine digitale Entität (Wurzelzertifikat)
  - gegenseitige Anerkennung paralleler CA-Hierarchien
  - erfordert zentrale Instanz zur Verwaltung der TSL (Bridge CA)

# Gültigkeit

## Schalenmodell

- Signatur ist gültig, wenn das zugehörige Zertifikat zum Zeitpunkt der Signaturprüfung gültig ist

- alle Zertifikate im Zertifizierungspfad müssen zum Verifikationszeitpunkt gültig sein
- alle übergeordneten Zertifikate müssen mindestens den gleichen Gültigkeitszeitraum haben
- einfache Verifikation, aber lange Gültigkeit erforderlich

### Kettenmodell

- Signatur ist gültig, wenn das zugehörige Zertifikat zum Zeitpunkt der Signaturerstellung gültig ist
- Nachteil: Ausstellungszeitpunkt muss auf geeignete Weise festgehalten werden

## Signaturniveaus

### Einfache Elektronische Signatur

- keine Sicherheitsanforderungen
- Daten, die mit anderen Daten verknüpft sind und zur Authentisierung dienen
- Beispiel: eingescannte Unterschrift beim Paketdienst

### Fortgeschrittene Signatur

- eindeutig dem Unterzeichner (natürliche Person) zugeordnet
- ermöglicht Identifizierung des Unterzeichners (über Zertifikat)
- ist mit Mitteln erzeugt, die Unterzeichner unter alleiniger Kontrolle verwenden kann (geheimer Schlüssel)
- nachträgliche Veränderung der Daten ist erkennbar (zB Signatur des Hashwertes)
- keine Sicherheitsanforderungen
- Beispiel GPG

### Qualifizierte Signatur

- beruht auf qualifiziertem Zertifikat, das zum Zeitpunkt der Signaturerstellung gültig war (z.B. Ausgabe durch qualifizierten Vertrauensdienstanbieter)
- ist mit qualifizierter elektronischer Siganturerstellungseinheit erzeugt

→ rechtlich äquivalent zur handschriftlichen Unterscheid, soweit diese nicht explizit im Gesetz gefordert ist

# ▼ Entwurf sicherer Systeme

## Security Engineering

- IT-Sicherheit erfordert planvolles und ingenieurmäßiges Verhalten
- schrittweise Konkretisierung und Verfeinerung
- ähnliche Schritte wie beim Software-Engineering
- Kombination formaler und informaler Werkzeuge (Modellierung, Analyse, Verifikation, Validierung, Tests)

→ Sicherheitsprüfung muss vor der Implementierung sein, damit man sicherheitsrelevante Aspekte und Sicherheitsgarantien schon in die Entwicklung einfließen lassen kann

→ Risikoanalyse vorher ist häufig möglich

→ muss in den Prozess integriert werden

## Allgemeine Prinzipien nach Saltzer und Schroeder

- Least-Privilege (Need to know)
  - jede Subjekt sollte nur das wissen und nur Privilegien haben, die zur Durchführung unbedingt benötigt werden
- Fail-safe default
  - Zugriffe grundsätzlich verboten und müssen explizit erlaubt werden
- complete mediation
  - jeder Zugriff muss auf Zulässigkeit geprüft werden
  - Umgehen der Zugriffskontrolle nicht möglich
- economy of mechanism
  - Sicherheitsmechanismen sollen einfach und klein sein
- open design
  - Architektur der Sicherheitssysteme sollte nicht geheim sein → kein Security by Obscurity

- separation of privilege
  - es ist sicherer, Zugriff auf kritische Objekte vom Zusammenwirken mehrer Bedingungen abhängig zu machen
- least common mechanism
  - Zahl gemeinsam genutzter Objekte minimieren
  - gemeinsame Objekte bieten Gefahr zwischen unerwünschtem Informationsfluss
- psychological acceptance
  - Mechanismen müssen einfach benutzbar sein
  - Zugriff genauso leicht wie ohne Sicherheitsmechanismen

## Security Development Lifecycle von Microsoft

- Tipps für jeden Schritt im Lebenszyklus einer Anwendung
- Tooluntersützung für viele Schritte

1. Training → Security Training
2. Requirements → Sicherheitsanforderungen herausarbeiten
3. Design → Designanforderungen, Threadmodellierung
4. Impelementation → approvete Tools nutzen, unsichere Funktionen ersetzen
5. Verification → dynamische Analse, Fuzz Testing
6. Release → incident response Plan erstellen
7. Resume → Wartung, Bugs beheben, Incident response

→ auch Angebote anderer Hersteller

## Tools: Statische Analyse

- z.B. gcc -Wall oder lint
- Code wird auf typische Fehlerkonstruktionen abgesucht
  - gute Programme berücksichtigen Kontext
  - teilweise werden Beispieleingaben oder -abläufe erzeugt
- sollte dauerhaft im Entwicklungsprozess integriert sein

    - sonst zu viele Fehler auf ein mal
- Beispiele: clang, eclipse…

## Tools: Dynamische Analyse

- Programm wird bei Ausführung beobachtet und ggf. extra in schwierige Situationen gebracht
- Programm wird instrumentiert, um es beobachten zu können → Welche Pfade werden verwendet? Was passiert, wenn Bibliotheksfunktionen Fehler zurückgeben?
- kann nur Fehler in ausprobierten Pfaden finden
- Beispiele: Valgrind, Purify, Unit Tests und Assertions

## Tools: Fuzzing

- Programm mit zufälligen Eingaben versehen und beobachten
- aber: man muss Input bauen, der zufällig ist und Regeln missachtet, aber gleichzeitig so regelkonform ist, dass Programm nicht direkt abbricht
    - Komponenten:
        - Eingabegenerator → Wie viel Zufall? Wie viel Protokoll?
        - Lebendigkeitsprüfung → Wann ist Kandidat abgestürzt?
        - Protokollierung → Was wurde geschickt, um Absturz auszulösen?
- meist eigenständiges Programm nötig
- fertige Programme für Dateien, TCP-Verbindugen, Webdienste

## Automatische Beweise/Formale Verifikation

- automatische Beweise mit Hoare Kalkül
- Vor- und Nachbedingungen müssen gegeben sein
- Programm sucht Beweiskette:
    - gelegentlich Hilfe nötig (z.B. Invarianten)
    - ohne Gegenbeispiele schwer zu unterscheiden, ob Beweiser keinen Weg gefunden hat oder ob Software Fehler hat

- ähnlicher Ansatz sind Model Checker → Systembeschreibung M und logische Eigenschaft $\varphi$ und man prüft, ob $M \models \varphi$

## Zertifizierung am Beispiel Common Criteria

- internationaler Standard zur Evaluation und Zertifizierung von Software
- man weist nach, dass man sich an bestimmten Sicherheitsprozess gehalten hat
- 7 Stufen der Vertraulichkeit:
  - von funktional getestet bis formal verifiziert
- verschiedene Funktionsklassen:
  - können frei gewählt werden

**Vorteil**

- Nachweis, ohne dass jeder Kunde selbst evaluiert

**Nachteil**

- aufwändig
- unflexibel

# ▼ Netzwerksicherheit

- grundsätzlich gibt es die gleichen Ziele und Probleme wie in lokalen Setups
- was ist anders?
  - höherer Schaden möglich → erfolgreiche Angriffe kompromittieren auch andere Systeme
    - ein Exploit reicht für viele gleichartige Systeme
  - höhe Bedrohung → mehr potentielle Angreifer (alle Nutzer im Internet)
    - geringere Hemmschwelle, da schwieriger nachvollziehbar
    - einfacherer Zugang
  - mehr Schwachstellen → grundsätzlich nicht, aber:
    - zusätzliche Protokolle mit eigenen Angriffspunken
    - zusätzliche Angriffsarten

## Internetworking

- verschiedene Netze, die teilweise miteinander interagieren

## TCP

- was genau bei TCP-Ports gesprochen wird, weiß man nicht → man muss bei Port 80 nicht HTTP sprechen

## Angriffe im Netzwerk

- Vertraulichkeit:
  - Daten von Servern unbefugt abrufen
  - Kommunikation mitschneiden
- Integrität:
  - Daten auf Systemen ändern
  - Kommunikation verfälschen
- Verfügbarkeit
  - Kommunikation zwischen Server und Client unmöglich machen

## Angriffsarten

- unabhängig vom Netzwerk
- bereits beschriebene Schwachstellen wie Buffer Overflow, SQL Injection → Datenabfluss, Manipulation, interaktiven Code ausführen
- Unterscheidung: mit oder ohne Abmeldung
- Fehlkonfigurationen: Default-Zugangsdaten oder interne Dienste, die extern erreichbar sind

## Angriffe auf Vertraulichkeit

- Internet/Netzwerk mit verschiedenen Verantwortlichkeiten
  - Kompromittierung von Routern oder Leistungsstrecken → insbesondere allerletzte Meile (WLAN im Cafe o.ä.)
  - Fehlkonfiguration von Routern (Umleiten des Verkehrs)

- lokale Netzwerke
  - Kompromittierung anderer, aktiver Komponenten
  - physischer Zugang zum Kabel
  - Umleiten des Verkehrs

### Beispiel LLMNR+WPAD+SMB

- Ausgangslage: Windows PC ohne Änderungen
- LLMNR - link local mutticast name resolution:
  - scheitert eine Namensauflösung per DNS, wird versucht, den Namen lokal P2P aufzulösen
  - Multicast-UDP-Paket an alle im gleichen Netz, d.h. alle dürfen antworten
  - WPAD (web proxy autodiscorvery protocol): sucht Proxykonfiguration für den Browser
  - Angriff:
    - Angreifer mit Zugriff auf gepatchte Netzwerkdose / Rechner im Netz
    - startet Webserver mit WPAD.DAT, darin ist der eigene Rechner als Proxy definiert
    - antwortet auf LLMNR-Fragen nach wpad.* mit eigener IP
    - startet SMB-Dienst, der eine Anmeldung mit NTLMv1 erfordert
    - → viele NTLMv1-Passwort Hashes

## Angriffe auf Integrität

- ähnlicher Ablauf wie bei Angriffen auf Vertraulichkeit
  - typischerweise aber aufwändiger, da nicht nur eine Kopie reicht
  - echte Nachricht muss unterdrückt werden
  - gefälschte Nachricht muss nachvollziehbar eingepasst sein (Paketcount, Prüfsummen,…)
- unbefugt Aktionen oder Änderungen remote ausführen:
  - keine Anmeldung vorgesehen
  - Anmeldung kann technisch umgangen werden

    - triviale Zugangsdaten
    - Manipulationsmöglichkeit über andere Netzwerkanwendung

## Angriffe auf Verfügbarkeit

- verschiedene Anknüpfungspunkte
    - Clientsystem, Clientanwendung (Absturz, Systemüberlast)
    - Anbindung des Clients (Sättigung der Leitung)
    - innerhalb des Internets (Fehlleitung)
    - Anbindung des Servers (Sättigung der Leitung)
    - vorgelagerte Systeme (Load Balancer, Firewall,…)
    - Serversystem, Serveranwendungen
    - Backendsysteme und -anwendungen

## Denial of Service

- Verfügbarkeit kann auf zwei Arten angegriffen werden:
    - regulären Programmfluss stören
        - Serverprogramm oder OS hängt oder stürzt ab
        - idR durch Programmierfehler im Programm ausgelöst
    - sinnlos Ressourcen belegen
        - fehlerhafte Anfragen des Angreifers belegen Ressourcen, damit reguläre Anfragen nicht bedient werden können
        - mögliche Ressourcen: Bandbreite der Anbindung, Speicher, Rechenzeit, künstliche Beschränkungen, Geld
        - Server nach Angriff idR wieder verfügbar

### SYN-Flood

- erzeugt eine große Anzahl von SYN-Paketen (Verbindungsaufbau) mit gefälschten IP-Adressen (Rechner, die nicht existieren oder antworten)
- wartet auf Bestätigung gemäß 3-Way-Handshake, Empfangsfenster voll, blockiert

- Flooding Angriffe können auch auf andere Weise z.B. mit ping durchgeführt werden

### DNS Reflection

- Beispiel für dDoS
- Antworten auf DNS-Anfragen sind meist länger als Anfragen selbst
- Absender kann gefälscht werden und das Opfer bekommt ungebetene Antworten → dort verstopfen sie meist bereits die externe Leitung

## Informationsgewinnung für Angriffe

- jeder Angreifer braucht Infos über
  - Name der Maschine
  - Adresse der Maschine
  - Prozessortyp
  - OS
  - Netzwerkdienste
  - Anwendungssysteme
  - Benutzer und Passwörter

→ je mehr er weiß, desto leichter der Angriff

→ Konflikt, denn viele dieser Infos werden für stabile Kommunikation benötigt

## Informationsquellen

- Informationsdienste
  - netstat (Infos über TCP/UDP-Verbindungen)
  - ping (Erreichbarkeit von Rechnern)
  - whois (Infos über Domains)
  - RIPE-Datenbank (Infos über vergebene IP-Adressen)
- Fehlermeldungen → oft sehr aussagekräftig
- Portscanner → Abfrage der auf einer konkreten Maschine offenen Ports

# Firewall

- Netzwerksicherheitsvorrichtung, die eingehenden aus ausgehenden Netzwerkverkehr überwacht und auf Grundlage von Sicherheitsregeln entscheidet, ob Datenverkehr zugelassen oder blockiert wird
- gesamter Datenverkehr von draußen nach drinnen und umgekehrt wird durch Firewall geschützt
- nur autorisierter Datenverkehr definiert durch lokale Sicherheitsstrategie darf durch
- Firewall muss unverletzlich sein ggü. Angriffen
- kann Software, Router oder Rechnerknoten sein, virtuell als auch physisch
- physische Isolation vom Internet ist sicher, aber meistens nicht gewollt
- Personal Firewalls zur Absicherung einzelner Rechner
- Firewall muss unverletzlich sein gegenüber Angriffen

# Varianten nach Ort

- lokale Firewall
  - überwacht nur lokal ein- und ausgehenden Datenverkehr
  - sichert Rechner nur gegen Netz ab, hilft nicht gegen kompromittierte Rechner
- Firewall am Netzübergang
  - überwacht Datenverkehr am Übergang zwischen zwei Netzen
  - sichert nur gegenüber Rechnern in anderen Netzen ab → hilft nicht gegen kompromittierte Rechner im selben Netz
- Bastion Host
  - Rechner, der das zu sichernde Netz nach außen repräsentiert
  - muss als exponierter Angriffspunkt besonders gesichert sein
  - gleichzeitig ggf. einziger Rechner, der Netz nach innen repräsentiert
  - üblicherweise Plattform für Circuit- und Application Level Gateways

# Varianten nach Schicht

- Paket-Filter (ISO Schicht 3 und 4)
  - einfache Filterung von einzelnen Datenpaketen
  - abhängig von Typ, Adresse und Ziel wird ein Paket durchgelassen oder nicht
  - kann nichts mit Content entscheiden
- Stateful-Packet-Inspection-Firewall SPI (ISO Schicht 3 und 4)
  - dynamische Paketfilterung → aktive Verbindungen werden mitgeschrieben!
  - Filterung basiert auf Vergleich und Korrelation mehrerer Datenpakete
  - Verbindungen werden in dynamischen Zustandstabellen gespeichert
  - DoS-Schutz möglich
- Circuit Level Gateway (ISO Schicht 6)
  - Mittler und Filter zwischen Client und Server
  - baut eigene Verbindungen zum Quell- und Zielsystem auf
  - keine Filterung von Inhalten
  - z.B. SOCKS-Server, der prüft, ob eine TCP-Verbindung erlaubt ist
- Application Layer Gateway ALG (Proxy-Firewall) (ISO Schicht 7)
  - Mittler und Filter zwischen Client und Server bez. einer bestimmten Anwendung (z.B: WWW, FTP, SMTP)
  - baut eigene Verbindungen zum Quell- und Zielsystem auf
  - überprüft auch Netzwerkpakete und übertragene Nutzdaten → Verschlüsselung an Firewall wird aufgebrochen!
  - Proxy-Server und -Client bilden eine Illusion des realen Server oder Nutzers
- Web-Application Firewall WAF (ISO Schicht 7)
  - ALG für HTTP
  - schützt Webapplikationen vor einer Vielzahl verschiedener Bedrohungen
  - überprüft eingehenden Web-Traffic
- Next Generation Firewall (ISO Schicht 3-7)

    - standardmäßige Firewall-Funktionen wie Stateful Inspection
    - Deep-Packet-Inspection (Inhalt konkret anschauen)
    - Entschlüsseln von TLS-Verbindungen
    - Application-Control → potentiell riskante Anwendungen können sichtbar und blockiert werden
    - integriertes Intrusion Detection and Prevention System
    - Upgrademöglichkeiten zur Integration künftiger Informationsfeeds → Threat Intelligence
    - Techniken, die Reaktion auf neue Bedrohung ermöglichen

## Paketfilter

- kein hohes Sicherheitsniveau
- relativ einfache Administration
- geringe Investitionskosten → kostenlose Software unter verschiedenen OS vorhanden
- keine wesentliche Einschränkung des maximalen Datendurchsatzes
- einfache und grundlegende Absicherung
- kann theoretisch auf zu schützendem Rechner eingerichtet werden
- filtern eingehen sowie ausgehenden Netzwerkverkehr

## Firewalling mit Paketfilter

- leicht umzusetzen, da oftmals Zusatzfunktion eines Routers z.B. über ACL-Listen
- man kann auch einen Rechner als Paketfilter verwenden → iptables
- Filterung wird durch Regeln gesteuert:
    - verbotsorientiert: alles, was nicht verboten ist, ist erlaubt → default permit
    - erlaubnisorientiert: alles, was nicht erlaubt ist, ist verboten → default deny
- Zustand
    - stateless: jedes Paket wird unabhängig behandelt

    - stateful: Kontextinformationen für Pakete derselben Session
- Filterkriterien
    - Richtung
    - Quell- und Zieladresse
    - Protokoll

## Firewallarchitektur P-A-P Struktur nach BSI

- Paketfilter - Application-Level-Gateway - Paketfilter
- Grundlage für hohes Sicherheitsniveau
- komplex, da mehrere Module verwendet werden
- nicht überall einsetzbar: z.B. kann IPSEC-Verkehr nicht über TCP/IP-Proxy geleitet werden
- einfache Erweiterungsmöglichkeiten, z.B. Anschluss von Virenscanner oder Spamfilter
- umfangreiche Protokollierungsmöglichkeiten

## Gegenmaßnahme: Verschlüsselung mit TLS

- Zwischensicht zwischen TCP und Anwendungsprotokoll
- Schutzziele:
    - Vertraulichkeit der übertragenen Daten
    - Authentizität des Servers (häufig)
    - Authentizität des Clients (sehr selten)
    - Integrität der übertragenen Daten
- richtige Einrichtung und Kontrolle nötig

## Werkzeuge zur Schwachstellenanalyse

- testen auf bereits bekannte und dokumentierte Sicherheitslücken
- finden kaum Fehler in Konfiguration oder unbekannte Fehler
- können gelegentlich zur Analyse auf eigenen Netze angewendet werden

- z.B. Metasploit → Prüfung vorhandener Netzwerkdienste, Prüfung auf sichere Konfiguration, Prüfung auf vorhandene bekannte Schwachstellen

## Penetration Test

- Angriff wird simuliert, um Angriffsvektoren zu identifizieren und Verteidigungsmaßnahmen auf Wirksamkeit zu untersuchen
- kann stark automatisiert oder individuell per Hand durchgeführt werden
- je mehr automatisiert, desto mehr, aber desto weniger spezifisch kann getestet werden
- beschränkte Ressourcen (vor allem Zeit)
- Ergebnis ist stichprobenartige Überprüfung des Sicherheitskonzeptes

## Packet-Sniffer (Software)

- Programme zur Analyse des Netzverkehrs
- gehören zum Standardumfang eines OS (z.B. tcpdump)
- erlauben Inhaltsanalye von Datenpaketen, die an Netzwerkkarte lesbar sind
- idR Adminrechte erforderlich
- nur Pakete des Ethernet Segmentes sind lesbar, in dem sich der Sniffer befindet (zB am selben Hub)
- gefährlich bei Diensten, die Authorisierung ohne Verschlüsselung durchführen
- dient insbesondere zum Ausspähen von Passwörtern

## Packet Sniffer (Hardware)

- Networttabs
- in die Verbindung eingeschliffen (Kupfer- oder Galsfaserkabel)
- manuelle Analyse des Datenverkehrs z.B. mit Wireshark

## Intrusion Detection and Prevention

- Wie kann man feststellen, ob in ein System eingedrungen wurde/wird?

- Anomalieerkennung: definiere normales Verhalten im System und erkenne signifikante Abweichungen
- Angriffserkennung: charakterisiere typische Angriffe auf das System als Angriffsmuster und erkenne Auftritt eines Musters

# Formen der Anomalieerkennung

- hostbasierte Detektionssysteme
  - HIDS (host based IDS)
  - HIPS (host based IPS)
  - werden auf Client ausgeführt und detektieren Datenverkehr zwischen Client und Netzwerk
- netzwerkbasierte Detektionssysteme
  - NIDS (network based IDS)
  - NIPS (network based IPS)
  - werden verteilt im Netz ausgeführt, um angeschlossene Clients und Server vor Eindringlingen zu schützen

# Anomalieerkennung erklärt

## Statische Anomalieerkennung

- geht davon aus, dass es statische Teile im System gibt
- Status der Dateien wird in Form von Signaturen (hashing) abgelegt
- periodisch werden die Signaturen neu berechnet und mit gespeicherten verglichen → bei Abweichung unzulässige Änderung

## Dynamische Anomalieerkennung

- Aufbau eines Normalprofils für einzelne Benutzer in mehreren Dimensionen
  - Loginzeit, Loginort, verwendete Programme
  - Betriebsmittelverbrauch
  - typische Aktionsfolgen
- Analyse meist statisch

- Einsatz von Clusteranalyse

# Angriffserkennung

- Regelbasierte Systeme
  - Angriffsmuster sind Satz von Regeln
  - Systemzustand ist in Wissensbasis abgelegt
    - Faktenbasis: Systemzustand in Form von Monitoring-Dateien und Audit-Records
    - Regelbasis: Bekannte Angriffsmuster in der Form Prämisse → Konklusion
- Vorgehensweise
  - Prüfung, ob irgendwelche Prämissen erfüllt sind
  - Prüfung, ob alle Prämissen dieser Regel erfüllt sind
  - Sammlung aller anwendbaren Regeln
  - Konfliktauflösung → Auswahl der besten Regel
  - Anwendung der Regel

# Protokollierung und SIEM

- je mehr System, desto mehr Daten, die man auswerten muss
  - Firewall
  - Router
  - Switches
  - Server
  - Clients

→ regelmäßige Auswertung erforderlich

- Protokollierungsserver (kleine Infrastruktur)
- Security Information Event Management (SIEM) Systeme ab mittelständischen Unternehmen

## Was ist ein SIEM?

- Kombi aus Security Event Management (SEM) und Security Information Management (SIM)
- SEM
  - zentralisiert Speicherung und Interpretation gesammelter Daten
  - erlaubt Analyse der Daten in nahezu Echtzeit
- SIM
  - Echtzeitüberwachung
  - Korrelation von Ereignissen
  - Benachrichtigung und Darstellung

→ SIEM beschreibt die Fähigkeit, Protokolldaten diverser Systeme zu sammeln, zu verknüpfen, zu analysieren und darzustellen, um daraus Sicherheitsmaßnahmen abzuleiten

# ▼ IT-Forensik

- Nachweis und die Ermittlung von Straftaten im Bereich Cyberkriminalität
- digitale Kriminalistik

## Ziel

- Erkennen von Methode oder Schwachstelle, die zum Systemeinbruch geführt hat
- Ermittlung des entstandenen Schadens
- Identifikation des Angreifers
- Sicherung der Beweise

⇒ so viele Informationen wie möglich von einem kompromittierten System sammeln, ohne den aktuellen Zustand bzw. Status des Systems zu verändern

## Computer-Kriminalität

- Delikte, rechtswidrige oder schädliche Verhaltensweisen, bei denen Computer Werkzeug oder Tiel der Tathandlung ist

# Schwerpunkte der Ermittlung

**Computer als Ziel**

- Aufklärung genutzter Schwachstellen
- Aufklärung der Tat → sind Daten abgeflossen oder verändert worden?

→ Wiederherstellung des Betriebs, Schwachstellen schließen, Betroffene informieren

**Computer als Werkzeug**

- Nachweis der Tat und des Täters
- gerichtsverwertbare Dokumentation

→ Aufklärung des Vorgangs und Strafverfolgung

# Anforderungen an den Ermittlungsprozess

- Akzeptanz: Schritte und Methoden müssen akzeptiert und gültig sein
- Glaubwürdigkeit: Funktionalität und Robustheit der Methoden kann nachgewiesen werden
- Wiederholbarkeit: Ermittlungsprozess kann wiederholt werden
- Integrität: Spuren dürfen nicht verändert werden
- Ursache und Auswirkungen: nachvollziehbare Verbindungen zwischen Personen, Ereignissen und Spuren müssen nachgewiesen werden
- Dokumentation: alles angemessen dokumentiert

## Bweismittelsicherung

- Sachbeweis vs. Personenbeweis
- jede Aktion am System muss dokumentiert werden
- Beweismittelsicherung: lückenlose Beweiskette, keine Tätigkeiten an Beweismitteln ohne Zeugen, Beweisspuren vor Manipulation schützen

# Angriffsformen

- Identitätsdiebstahl/Phishing: Abfangen von Zugangsinformationen oder Aneignen persönlicher Merkmale

- Ransomware: Schadsoftware mit Ziel der Lösegelderpressun
- DDoS: Störung der Verfügbarkeit von Onlinediensten durch Ausschöpfen von Kapazitätsgrenzen
- Botnetz: gekaperte Rechner, die für kriminelle Zwecke missbraucht werden
- Advanced Persistant Threat: langfristige Spionage- oder Sabotagetätigkeiten, tw. staatlich motiviert

## Incident

- Störung im IT-Betrieb unabhängig von Auswirkung
- Security Incident: Verletzung eines Sicherheitsziels der Organisation, unterschiedliche Auswirkungen
  - Störung des Betriebs durch Ausfall oder Manipulation
  - Incident ohne Störung
    - Passwort Phishing
    - Malware Fund, z.B. trojanisierter Rechner
    - meldepflichtiger Datenabfluss

## Störung

- Prozesse oder Ressourcen stehen nicht wie vorhergesehen zur Verfügung
- innerhalb des Normalbetriebs behebbar

## Notfall

- erhebliche Unterbrechung eines zeitkritischen Geschäftsprozesses
- Notfallpläne

## Krise

- massive Unterbrechung  eines zeitkritischen Geschäftsprozesses
- Notfallpläne greifen nicht ausreichend oder liegen nicht vor

## Charakteristika eines Angriffs

- Infiltration: Auf welchem Weg kommt Angreifer?

- Exfiltration: Auf welchem Weg werden Daten herausgeschleust?
- Angriffswaffe: Welche Software wird genutzt?
- Angreifertyp: Ressourcen, Interesse und Strategie des Angreifers