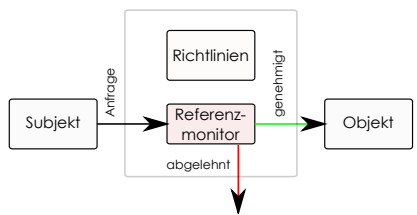
**Einführung**

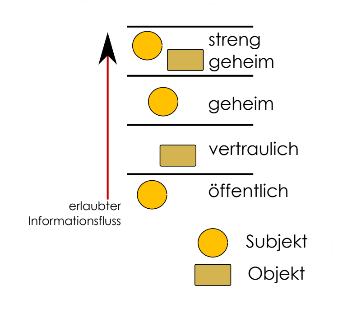
* Internet der nächsten Generation: hochgradig verteilt, vernetzt, IoT 🡪 Durchdringung des Alltags mit IT
* Begriffe:
  + Verwundbarkeit Schwachstelle des Systems, Umgehung der Sicherheitsrichtlinien
  + Angriff Versuch der Umgehung der Sicherheitsrichtlinien
  + Bedrohung Angriffsmöglichkeiten eines Systems
  + Risiko = Wahrscheinlichkeit für Eintritt des Schadens \* potenzieller Schaden
* Angriffsklassen
  + Benutzer & Daten Phishing, Spamming, Social Engineering
  + Anwendungen XSS, SQL-Injection, LDAP-Injection
  + Systeme Buffer-Overflow, Viren, Würmer, Trojaner
  + Netzwerke Sniffen, Spoofen, DoS
* Definition von Schutzzielen wichtig, um abstraktes Ziel Informationssicherheit zu konkretisieren
  + Hauptschutzziele:
    - **Integrität** Schutz vor unautorisierter & unbemerkter Modifikation von Daten
      * Vorgabe von Zugriffsrechten
      * Isolierung
      * Manipulationserkennung (Prüfwerte, digitales Watermarking)
    - **Vertraulichkeit** Schutz vor unautorisierter Informationsgewinnung
      * Regeln für (un-)zulässige Informationsflüsse
      * Verschlüsselung von Daten
      * Klassifizierung von Objekten & Subjekten
    - **Verfügbarkeit** Schutz vor unbefugter Beeinträchtigung der Funktionalität
      * Festlegen von Schwellenwerten
      * Hochverfügbarkeitslösungen (Hard- und Software)
      * Backup
  + Andere Schutzziele
    - Verbindlichkeit Schutz vor unzulässigem Abstreiten durchgeführter Handlungen
      * Festlegung welche Aktionen verbindlich sind
      * Protokollieren von Aktionen und Zeitpunkten
      * Beweissicherungen durchführen
    - Authentizität Nachweis der Echtheit und Glaubwürdigkeit der Identität
      * Regeln zur Vergabe von eindeutigen Identifikationen von Subjekten / Objekten
      * Verfahren zum Nachweis der Korrektheit der Identität
    - Privatheit Schutz der personenbezogenen Daten
      * Regeln zu Datenvermeidung und Datensparsamkeit
      * Festlegen von Verfallsdaten
      * Zweckbindung der erhobenen Daten
* Maßnahmen strukturiert in
  + Vermeidung und Verhinderung
  + Erkennung
  + Schadensbegrenzung

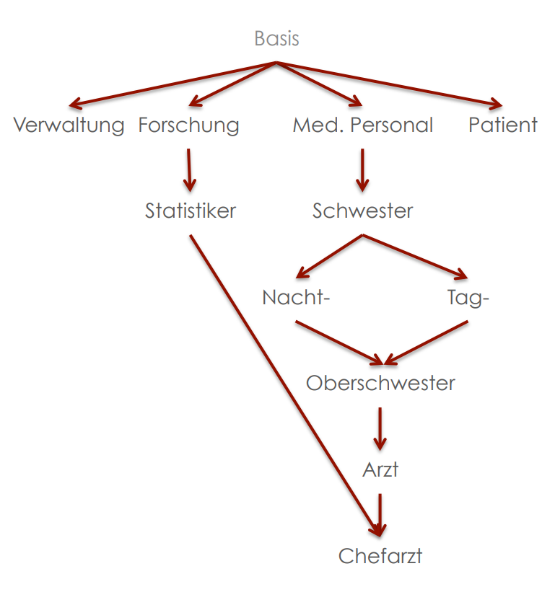
**Sicherheitsmodelle**

* Komponenten
  + Subjekt aktive Einheit, initiiert Zugriff auf Ressourcen (Person, Programm, ...)
  + Objekt soll geschützt werden, Informationen oder Ressource (Daten, Drucker, ...)
  + Referenzmonitor
    - nicht unbedingt als physikalische Einheit im System
    - kontrolliert Zugriffsversuche, logged Zugriffe
    - muss vor Manipulation geschützt werden
  + Richtlinie
    - definiert Bedingungen, unter denen Subjekt auf Objekt zugreifen darf
    - Beschreibt erwünschte, zulässige Zustände

Zugriffskontrolle

* Discretionary Access Control = DAC
  + benutzerbestimmte Zugriffskontrolle
  + Eigentümer ist selbst für Schutz eines Objekts verantwortlich
  + individuelle Rechtvergabe für Objekte, objektbezogene Sicherheitseigenschaften
  + Problem: keine Betrachtung von Abhängigkeiten, implizite Vergabe von Leserechten durch Ausführung einer Aktion
* Mandatory Access Control = MAC
  + systembestimmte Festlegung von Sicherheitseigenschaften
  + systembestimmte Rechte > benutzerdefinierte Rechte
  + OS oder Programm muss spezielle Maßnahmen / Dienste bereitstellen:
    - Zugriffsmatrix Grundlage der Zugriffskontrolle in allen Standard-OS
      * Matrix, die Rechte eines Subjekts auf andere Subjekte / Objekte festlegt
      * Vorteile
        + einfach & intuitiv nutzbar, flexibel & feingranular
      * Nachteile
        + keine Rechtevergabe an Klassen mit Rechte-Vererbung
        + schlechte Skalierung: zu viele und dynamische Menge von Subjekten



* + - Bell-La Padula-Modell
      * Zugriffsoperatoren read, write, exec, append, control
        + systembestimmte Regeln wie no-read-up oder no-write-down über verschiedene Hierarchien
      * Probleme wie blindes Schreiben möglich, keine Integrität
      * Teil von umfassenden Sicherheitsregularien, einfach zu implementieren
    - Role-Based Access Control (RBAC)
      * Rolle für bestimmte Aufgabe und damit verbundene Berechtigungen
      * Erfüllt Prinzipien need-to-know und separation-of-duty
      * verbreitet in ERP- (z.B. SAP) und CMS-Systemen
      * Ziel:
        + Vereinfachung von Verwaltungsaufgaben
        + Nachbilden hierarchischer Organisationsstrukturen
      * Subjekt kann nur in Rollen aktiv sein, in denen sie Mitglied ist, und besitzt immer nur die Rechte der aktiven Rolle
      * Aufgabentrennung
        + Statisch wechselseitiger Ausschluss von Rollenmitgliedschaften

Kein Subjekt ist Mitglied in mehreren Rollen (z.B. Kassierer & Kassenprüfer)

* + - * + Dynamisch wechselseitiger Ausschluss von Rollenaktivitäten

Kein Subjekt ist in mehreren Rollen gleichzeitig aktiv (Kontoinhaber & Kundenbetreuer)

* + - * Fazit
        + Sehr flexibel verwendbar, skalieren gut
        + Direkte Nachbildung von bekannten Organisations- und Rechtestrukturen
        + Intuitive und einfache Abbildung der Rollen auf Geschäftsprozesse
        + Einfache und effiziente Rechte-Verwaltung
    - Weitere Modelle:
      * Conflict of Interest Modelle
        + Zugriff auf Information abhängig von Mitgliedschaft zugreifender Subjekte in Klassen mit kollidierendem Interesse (z.B. Autohersteller <> Ölfirma)
      * Non-Interference Modelle
        + Effekte von Aktionen sind nur für Berechtigte sichtbar

**Schwachstellen**

Typen

* Konzeptionelle Schwachstellen
  + Keine ausreichende Klassifizierung von Information
  + Fehlende Rollenkonzepte & Sicherheitsregularien für Datenflüssen
  + Keine ausreichende Identifikation von Subjekten
  + Keine ausreichende Schulung der Mitarbeiter 🡪 Social Engineering
* Schwachstellen in der Konfiguration
  + Konfigurationsfehler
  + Firewall erlaubt gefährliche Kommunikation
  + Unsichere Verschlüsselungsverfahren erlaubt
  + Patchmanagement
* Schwachstellen in der Programmierung eingesetzter Software
  + Fehlende Validierung von Benutzereingaben
  + Unverschlüsselte temporäre Daten zur Laufzeit
  + Beliebige Fehler, die ein Programm abstürzen lassen

Keine ausreichende Identifikation von Objekten

* Es ist möglich, sich als jemand anderes auszugeben (**spoofing** = Verschleierung der Identität)
* Spoofing
  + ARP-Spoofing
    - Bei ARP-Broadcast zwischen verschiedenen Rechnern wird MAC-Adresse abgefragt
    - Wenn bösartiger Rechner statt dem Zielrechner schneller antwortet, wird Kommunikation mit diesem aufgebaut
    - Daten gehen an falschen Rechner
  + IP-Spoofing
    - Schadrechner möchte aus Internet Schadsoftware schicken, Firewall blockiert externe Absender
    - In IP-Paket wird zum Umgehen der Firewall eine interne IP-Adresse als Absender angegeben
    - Firewall lässt Paket durch
  + DNS-Spoofing
    - Bei Aufruf einer Homepage wird diese vom DNS-Server in IP-Adresse umgewandelt
    - DNS-Server unter Kontrolle des Angreifers liefert IP eines bösartigen Webservers zurück
    - User gibt Informationen auf bösartiger Seite Freitag
  + Web-Spoofing
    - User ist auf einer dubiosen Webseite, auf der eine Meldung angezeigt wird
    - User klickt auf „OK“, Schadsoftware wird heruntergeladen, User installiert sie nach Anweisung der Webseite
* Identitäten sind leicht fälschbar, deshalb starke Authentifizierungs- und Autorisierungsverfahren benötigt

Programmierfehler

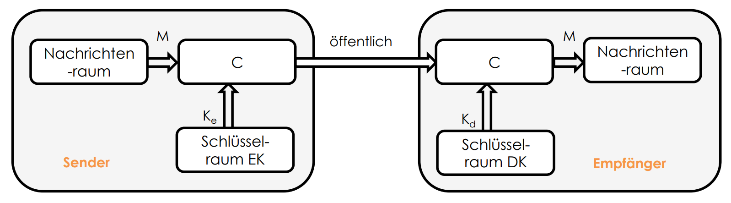
* All Input are evil 🡪 Software muss auch mit unerwarteten Eingaben umgehen können
* Angriffe:
  + Buffer Overflow
    - Überlauf des Speichers, mehr Daten in Speicherbereich als zugelassen
    - Rücksprungadresse wird überschrieben
      * Ungültiger Wert für Programmabsturz (DoS)
      * Rücksprungadresse kann auf andere Funktion des ausgeführten Programms zeigen
        + Ausgabe von Informationen die nicht für Subjekt bestimmt sind
      * Rücksprungadresse zeigt auf Anfang des überschriebenen Stacks
        + Ausführung von eventuell platziertem Binärcode 🡪 Code Injection
    - Mögliche Angriffsszenarien
      * Angriff von Innen Schwachstellen im Programm, die mehr Rechte haben als der Angreifer
      * Angriff von Außen Schwachstellen in Serverdiensten
    - Ziel Systemabsturz oder Erlangen von Informationen, die Angreifer nicht zugänglich sein sollten
  + Heap Overflow
    - Dynamisch allokierte Speicherblöcke, Überschreiben von Programmvariablen mit Usereingabe möglich
  + BSS-Overflow
    - Betrifft globale Systemvariablen, z.B. überschreiben vor Variablen für aufgerufene Unterprogramme
* Abwehrmechanismen:
  + Code und Audit
    - Secure Programming
      * Hardwarenahe Sprachen (C, C++) nur einsetzen wo nötig
      * Vermeidung von Befehlen, die die Eingabelänge nicht verifizieren
      * Manuelle Verifikation von Benutzereingaben
    - Code Audit
      * Qualitätssicherung durch von Programmierer unabhängigen Auditor
      * Automatisierte Code Audits durch Tools
    - Binary Audit
      * Identisch zu Aktionen von Angreifern, mit Ziel Schwachstellen zu identifizieren
      * Automaitisiert als Netzwerk- oder Hostbasierte Audits
  + Compiler und Library
    - Canary-Basierter Stack-Schutz
      * Zwischen gespeicherter Adresse und Variable wird ein Wert abgespeichert
      * Wenn dieser verändert wurde, wird das Programm abgebrochen
      * ABER: DoS funktioniert immer noch
    - Sicherung der Rücksprungadresse
      * Rücksprungadresse wird an weiterem Platz gesichert
      * Letzte Operation in Funktion schreibt Rücksprungadresse zurück
    - Safe Library
      * Normale Aufrufe unsicherer Funktionen werden mit Wrapper durch sichere ersetzt
  + Betriebssystem
    - Non-Executable Stack
      * Daten im Stack als nicht ausführbar markiert
    - Address-Space-Layout-Randomisation (ASLR)
      * Zufällige Vergabe der Adressbereiche von OS
        + Ermittlung der Rücksprungadresse erschwert
      * Unix- und Win-Kernel Standard

Web-Schwachstellen

* HTTP ist zustandsloses Protokoll, Cookies zum abspreichern von Benutzerinformationen
* Top Ten Schwachstellen
  + Cross Site Scripting (XSS)
    - Bösartiger Client greift anderen Client an, indem er ein Script auf dessen Rechner ausführt
    - Grundtypen
      * Stored
        + Schadcode auf Web-Server, i.d.R JavaScript in Benutzer-Browser
        + Nutzer-Eingaben werden ungeprüft an Browser anderer Nutzer gesendet
        + z.B. Script in Nutzereingabefeld, dass andere Nutzer aufrufen
      * Reflected
        + Eingabefelder (z.B. Suchfelder) spiegeln Eingabe des Nutzers zurück
        + Link mit Einträgen auf Eingabefeld wird an Opfer weitergeleitet
        + z.B. Script im Link, wo Eingabe dargestellt wird
      * DOM-Injection
        + Webserver verarbeitet Eingaben, die dem DOM mitgeliefert werden
        + Nicht sichtbar für den Nutzer, da Befehl im Code eingebettet ist
        + z.B. Verarbeitung einer PathVariable im JS-Code der HTML-Code erzeugt
    - Nutzer vertrauen Inhalten die ihr Browser ihnen zeigt
    - Gegenmaßnahmen
      * Auf Serverseite Parameter auf Metazeichen testen/filtern oder konvertierten
      * Auf Clientseite Scripte nur von vertrauenswürdigen Seiten erlauben und vorsichtig sein bei Links
  + (SQL) Injection
    - Manipulierte SQL-Eingabe in Webformular, Server erstellt daraus SQL-Befehl
    - SQL-Befehl wird auf Datenbank angewendet und greift mehr oder andere Daten ab als gewollt
    - Arten
      * Manipulation eines SQL-Befehls
      * Einschleusen eines Befehls, also Abschluss des Grundbefehls + Eingabe mehrerer danach
    - Ursache
      * Fehlende Eingabe-Prüfung / -Filterung / -Validierung
      * Programmierfehler
  + Weitere Schwachstellen
    - Broken Authentication
    - Sensitive Data Exposure
    - XML External Entities (XXE)
    - Brocken Access Control
    - Security Misconfiguration
    - Insecure Deserialization
    - Using Components with known vulnerabilities
    - Insufficient loggin and monitoring

**Verschlüsselung**

* Ziele:
  + Vertraulichkeit, Integrität, Authentizitä, Verbindlichkeit



Kryptographie:

* Anforderungen an kryptographische Verfahren:
  + Sicherheit nicht von Geheimhaltung der Ver- und Entschlüsselungsfunktionen abhängig
  + Geheimer Schlüssel nicht durch Kenntnis der verwendeten Verfahren berechenbar
  + Stärke des Verfahrens nur von Güte des geheimen Schlüssels abhängig (Kerckhoffs Prinzip)
* Brute Force der Schlüssel soll nicht möglich sein
* Verfahren
  + Symmetrisches Verfahren
    - Ver- und Entschlüsselungsschlüssel sind gleich oder einfach berechenbar
    - Gemeinsamer, geheimer Schlüssel (Secret Key) wird genutzt
    - z.B. ROT (Cäsar-Code), DES, AES
    - Problem ist sicherer Austausch des gemeinsamen Schlüssels K
  + Asymmetrisches Verfahren
    - Pro Kommunikationspartner ein Schlüsselpaar
    - Einwegfunktion mit Falltür zur Berechnung der Schlüssel
      * Invertierung der Funktion nur mit geheimen Schlüssel möglich
    - z.B. RSA, EIGamal-Verfahren
  + Hybride Verschlüsselung:
    - Nutzdaten symmetrisch Verschlüsselt, dafür nötige Schlüssel asymmetrisch verschlüsselt
    - Beide Kommunikationspartner müssen öffentliche & private Schlüssel besitzen
* Diffie-Hellman Schlüsselaustausch
  + Gegenseitiger Austausch von öffentlichen Schlüsseln & Berechnungen mit geheimen Schlüsseln führen zu schlussendlicher Berechnung des gleichen, geheimen Hauptschlüssels
  + Problem: kein Schutz vor Man-in-the-Middle-Angriffen
    - Also Fehlende Authentizität
    - Lösung: Hash-Wert und und Dokument werden über verschieden Kanäle getauscht, bei Veränderung stimmt Hash-Wert nicht mehr überein
      * Anforderungen:
        + Einfach berechenbarer Hash-Wert
        + Keine Bestimmung der Hashfunktion mithilfe des Hash-Werts möglich
        + Keine Bestimmung eines anderen Wertes mit dem selben Hash-Wert mithilfe des Grundsätzlichen Werts möglich
* Wenn alle Eigenschaften erfüllt, heißt es eine kryptographische Hashfunktion

Verschlüsselung und Authentizität

* Ohne Überprüfung der Authentizität ist Verschlüsselung nicht viel wert
  + Schutz vor Known-Ciphertext-Attacks gesucht
* MAC
  + Für Zuordnung verschlüsselter Nachricht an Besitzer
  + Gut für symmetrische Verschlüsselungsverfahren
* Digitale Signatur
  + Idee:
    - Hash einer Nachricht mit privatem Schlüssel in asymmetrischen Verfahren verschlüsseln
    - Jeder kann prüfen, dass Sender einen privaten Schlüssel besitzt
  + PKI = Public Key Infrastructur
    - Zuordnung von öffentlichen Schlüsseln zu Personen und Validierung dieser Zuordnung
    - Komponenten
      * Certification Authority (CA): stellt Zertifikate aus, signiert und veröffentlicht sie
      * Registration Authority (RA): bürgt für Verbindung zwischen öffentlichem Schlüssel und Identitäten
      * Validation Authority (VA): ermöglich validierung der Zertifikate
      * Verzeichnisdienst: Verteilung der Zertifikate
  + Arten digitaler Signaturen
    - Einfache Signatur
      * Keine speziellen Anforderungen an Zertifikate und Erzeugung
      * Geschädigter muss Schaden nachweisen
    - Fortgeschritten Signatur
      * Anforderung an Zertifikataussteller, Signierumgebung und Verknüpfung der Signatur mit Dateien
      * Signaturanbieter haftet für Richtigkeit und Vollständigkeit
    - Qualifizierte Signatur
      * Forgeschrittene Signatur mit qualifizertem Zertifikat
      * Sichere Signaturerstellungseinheit
      * Rechtlich gleichgestellt mit Unterschrift
  + Hybride Verfahren
    - Kombinieren Vorteile der Einzelverfahren
    - Standard für Verschlüsselung im Internet

Ziele

* Vertraulichkeit durch Schlüsselaustausch und Verschlüsselung
* Integrität durch Hashfunktion und MAC
* Verbindlichtkeit durch digitale Signatur, Zertifikate und PKI
* Authentizität durch MAC und Signaturen

**Systemsicherheit**

* Aspekte der IT-Systemsicherheit:
  + Analyse von möglichen Angriffsvektoren / Schwachstellen
  + Analyse der Ausbreitung von Angriffen
  + Analyse von Schadensszenarien
  + Entwicklung von Abwehrmaßnahmen

Malware

* **Mal**icious Soft**ware**
  + Nutzt Systemschwachstellen aus, zentrale Komponente der Mehrzahl von Angriffen
  + Hochentwickelte Technologie, verschiedenste Verbreitungswege
* Typen:
  + Virus
    - Ausführbarer Code, nistet sich in anderes Programm ein
    - Beinhaltet Infektions- und Schadteil
    - Nur aktiv wenn Wirt aktiv ist, Verbreitung durch Datenaustausch
    - Arten
      * 2. Generation von Viren befällt Addons, Plugins und Interpreten, als Vorbereitung des eigentlichen Angriffs
      * Makro- und Datenviren eingebettet in Dokumenten, werden durch Lesezugriff ausgelöst
      * Ani-Viren sind verseuchte Dokumente, die Schwachstellen im Interpreter ausnutzen
  + Würmer
    - Selbstständig ausführbares Programm, fähig zur Reproduktion
    - Verbreitet sich aktiv selbst
  + Trojanisches Pferd
    - Schadprogramm oder -Code tarnt sich als ordnungsgemäßes Programm
    - Installiert Schadcode häufig nach (z.B. Backdoors oder Spionagesoftware)
* Infektionswege
  + Downloads von Webseiten oder Emails
  + Soziale Netzwerke
  + Eigene Verbreitunsgmechanismen

Rootkit

* Sammelbegriff für einen Satz von Werkzeugen
* Ziele:
  + Erlangung von Root- oder Adminrechten
  + Verbergen der eigenen Aktivität durch Einschleusung von Malware
* Typen:
  + App-Rootkit Modifikation von Systemprogrammen
  + Kernel-Rootkit Modifikation von Kernel-Mode Code (z.B. Treiber)
  + Userland-Rootkit Modifikation von Usermode-Shared-Libs
  + Speicher-Rootkit Modifiziert RAM von laufenden Prozessen

Botnet

* Gruppe von Programmen, die auf Anweisung Befehle auf einer vernetzten Rechnergruppe ausführen
* Illegales Botnet:
  + Bots ohne Wissen und Zustimmung der Eigentümer auf Rechnern installiert
  + Versenden von Spam, DDoS, Proxy für illegale Inhalte…

**APT = Advanced Persistant Threats** 🡪 Angriffe, die über längere Zeit unentdeckt Daten extrahieren

Abwehmechanismen

* Reaktive Maßnahmen
  + Client: Virenscanner, Prüfung von Systemdateien auf Modifikation
  + Server:
    - Anti-Malware & Anti-Spam auf Email-, Anti-Malware auf File-Servern
    - Klassifizierung und Verbot von Downloads
* Proaktive Maßnahmen
  + Windows: Application Whitelisting, Automatische Updates, Client-Firewall
  + Unix: Partition mit ausführbarer Software read-only, User-Partition nonexec
  + Problem 🡪 kein absoluter Schutz möglich
* Weiterführende Mechanismen
  + Kontrolle von OS und Hardware außerdehalb des Systems
  + Definition sicherer Zustände und erlaubter Veränderungen
  + OS-Hersteller behalten Kontrolle (vgl. iOS, UEFI Boot, …)
  + Kryptographische Überprüfung ausführbarer Komponenten und Hardware-Schlüsselspeicher
  + Komplett neue Betriebssysteme

**Authentifizierung**

* Ziel:
  + Eindeutige Identifikation und Nachweis der Identität, Abwehr von Identitätsdibstahl
* Problem:
  + Nicht nur Mensch – Gerät, sondern auch Gerät – Gerät oder Gerät – Dienst Interaktion
* Personen, Geräte und Dienste müssten identifizert und authentifiziert werden
* Mehrfaktor Authentifizierung
  + z.B. 2-Faktor Authentifikation
  + einseitige oder wechselseitige Authentifizierung möglich

Authentifizierung durch biometrische Merkmale

* Merkmal muss universal, eindeutig und beständig sein (z.B. Fingerabdruck)
* Außerdem muss Merkmal performant erfassbar und fälschungssicher sein
* Unterscheidung in physiologische und Verhaltensmerkmale
  + Physiologisch: keine oder begrenzte Möglichkeit zur Auswahl oder Änderung (z.B. Gesicht)
  + Verhalten: Merkmal nur bei bestimmter Aktion vorhanden (z.B. Sprache)
* Vorgehen: Messdatenerfassung durch Sensor, Registrierung des Nutzers mit Daten
* Authentifizierung: Erfassen, Verifikation digitalisieren, mit Referenzwert vergleichen
* Probleme:
  + Angriffe: Täuschung des Sensors durch Attrape, Einspielen von Daten
  + Kopplung zwischen Merkmal und Person, Gefahr gewaltsamer Angriffe auf Personen
* Fazit
  + Nicht geeignet als ausschließliches Verfahren, idealer Einsatz als Mehrfaktor-Authentifizierung
  + Weiterentwicklung in Zukunft

Authentifizierung durch Wissen

* Passwortbasierte Zugangskontrolle
  + Eingabe von Benutzername und Passwort, anschließender Abgleich mit gespeicherten Daten
  + Statische oder Einmal-Passworte möglich
    - Statische Passwörter können gestohlen werden 🡪 unsicher
    - In sicheren Systemen sollte Passwort nur eine Komponente sein
  + PAP = Password Authentication Protocol
    - unverschlüsselte Übertragung von Passwort und Benutzerkennung

Protokolle

* Challenge Response Verfahren
  + Subjekt und Instanz haben gemeinsames Geheimnis, Subjekt muss Challenge von Instanz mithilfe von Geheimnis lösen, Instanz überprüft die Lösung
  + Einfache Authentifizierung über Netzwerke, Geheimnis muss nicht extra übertragen werden (Passwort)
  + z.B. CHAP
    - Subjekt sendet ID and Instanz
    - Intanz generiert Challenge bzw. Zufallszahl RAND und sendet diese an Subjekt
    - Subjekt berechnet Lösung R durch Hashing mit Passwort und sendet Ergebnis an Instanz
    - Instanz prüft Ergebnis
* Kryptographische gesicherte Authentifizierung
  + Nutzen Authentifizierungsserver, der geheimen Schlüssel mit allen Instanzen / Subjekten besitzt
  + Sicherheit des Verfahrens von Sicherheit der beteiligten Systeme abhängig, Authentifizierungsserver ermöglicht einfachen Betrieb
  + ABER: Authentifizierende Instanz muss Passwort wärhend des Prozesses besitzen
  + z.B. Needham-Schroeder Protokoll
    - A und B haben jeweils Schlüssel mit Authentifizierungsserver
    - A sendet ID von A, B und Nonce an AS
    - AS verschlüsselt Rückgabe mit Session Key und Paket für B mit Schlüssel von A und sendet zurück an A
    - A sendet Paket an B (beinhaltet Session Key)
    - B sendet Nonce verschlüsselt mit Session Key zurück an A
    - A beweißt Empfang der Nonce durch Ent- und Verschlüsselung + geheimer Operation
* Zero Knowledge Verfahren
  + Dritter soll Authentifzieren, ohne Kenntniss über Geheimnis zu erlangen
  + z.B. Feige-Fiat-Shamir Verfahren, gegeben 2 Primzahlen (p, q) und x=r2 mod n 🡪 gesucht r
    - Instanz berechnet x durch r und Vorzeichen, schickt x an Server
    - Server wählt Zufallszahlen und schickt an Instanz
    - Instanz berechnet y mit Zufallszahlen und r
    - Server überprüft y

**Autorisierung**

* Einsatzbereich:
  + Zugriff auf Ressourcen in einem Netzwerk, Installation und Benutzung von Software
* Generelle Lösung ist Zugriffsmatrix, wo die Rechte jedes Subjekts auf jede Datei inbegriffen sind
  + Zugriffskontrolllisten (Access List)
    - Vorteile
      * Rechte von Objekt effizient bestimmbar, Rechterücknahme effizient realisierbar
      * dezentrale Kontrolle möglich
    - Nachteile
      * Bestimmung von Subjektrechten sehr aufwändig
      * schlechte Skalierbarkeit bei dynamisch wechselnder Menge von Subjekten
  + Capability-Listen
    - Vorteile
      * Einfache Bestimmung von Subjekt-Rechten
      * Zugriffskontrolle einfach 🡪 nur Ticketkontrolle
    - Nachteile
      * schwierige Rechterücknahme
      * keine Subjekt-Ticket-Kopplung, Besitzt berechtig zur Wahrnehmung der Rechte
      * unübersichtliche Rechte für Objekt
  + Domain-Type-Enforcement
  + Lock-Key-Konzept
* Zugriffskontrolle
  + Unix
    - Angabe des Zwecks: r, w, x
    - Unix Kern überprüft ob Zugriff laut Access List (ACL) für Prozess genehmigt ist
    - am Ende Erstellung eines Filehandles (Capability) 🡪 in Capability List
  + Windows
    - Zugriffskontrolle mit Security Descriptoren, enthalten verschiedene Infos zu User und Rechten
    - Zugriff immer erlaubt, wenn keine ACL festgelegt ist
      * Subjekt hat automatisch Zugriffsrechte, wenn es Owner von Objekt ist
      * Security Descriptoren durchlaufen, IDs checken, wenn ID stimmt 🡪 Entscheidung ob access allowed oder denied

Kombination Authentifizierung und Autorisierung:

* Kerberos-Protokoll
  + Ziele:
    - Authentifizierung von Subjekten (Principals), Austausch von Sitzungs-Schlüsseln für Principals
    - Single-Sign-On für Dienste / Personen
  + Ablauf:
    - Nutzer gib ID und Passwort ein
    - Client sendet Daten an Authentication Server, bekommt verschlüsselte Daten zurück
    - Client sendet Daten an Ticket Granting Server, bekommt verschlüsselte Daten zurück
    - Client sendet Ticket an Server

**Kommunikationssverschlüsselung**

IPSec

* löst Sicherheitsprobleme von IPv4 und IPv6 durch Erweiterung des Headers
  + Transport Mode sichere Kommunikation zwischen Quelle und Ziel
    - IPSec header nach IP header, weil Kommunikation direkt zwischen Quelle und Ziel
  + Tunnel Mode Sicherung der Kommunikation nur zwischen Gateways
    - IPSec header nach outer IP header, weil Kommunikation zwischen Gateways
      * inner IP Header wird erst im Netzwerk gebraucht
* am Ende immer IPSec trailer
* Security Association (SA) Sicherheitsvorgaben für Kommunikation
  + bestimmt durch
    - Security Paramter Index (SPI)
    - Destination Address
    - Security Protocol Identifier
  + Datenbanken erforderlich für Verwaltung der Sicherheitsassoziationen
    - Association Database (SADB) enthält aktive SAs des Systems
    - Policy Database (SPD) gibt vor, für welche Datenströme SAs wie eingerichtet werden müssen
* Authentication Header (AH)
  + Sicherung von Authentizität und Integrität verbindungslos übertragener IP-Pakete
  + schützt gegen Spoofing, Modifikation des Inhalts und Replay-Attacken
  + HMAC-Algorithmus zur Verschlüsselung der unveränderlichen Teile

HMAC zur Verschlüsselung des Headers, IP-Datenteils und des Trailers

* Encapsulated Security Payload (ESP)
  + Vertraulichkeit der Übertragung und Authentizitätsprüfung
  + optionale symmetrische Verschlüsselung
  + optionale Authentifizierung durch HMAC-Algorithmus

Schlüsselmanagement

* ISAKMP (Internet Security Association and Key Management Protocol)
  + Protokoll zur Aushandlung von Sicherheitsparametern
  + Authentifizierung von Instanzen
* IKE (Internet Key Exchange)
  + Auf Diffie-Hellman basierendes Authentisierung- und Schlüsselaustauschprotokoll
  + Aushandlung von Sicherheitsassoziationen
  + Austausch einiger Informationen zwischen Initiator und Responder, benutzt Zertifikate zur Authentisierung

Transport Layer Security (TLS)

* Idee: Vertraulichkeit und Authentizität, länger gültig als eine Verbindung, verwendet Sitzungskonzept
* TLS Record
  + Berechnung eines MAC, Verschlüsselung der Daten und MAC
  + Fragmentierung und Komprimierung der übertragenden Daten
* TLS Handshake
  + Aushandlung von Sitzungsparametern
  + Sicherung der Konsistenz der Sitzungsinformationen
  + Austausch der Informationen, wie Key, Ciphers und Certificates
* Vorteile:
  + jedes Protokoll kann auf Basis von TLS impl. werden, Unabhängigkeit von App und System
* Nachteile:
  + rechenintensiver Verbindungsaufbau auf Serverseite
  + Authentifizierungs und Verschlüsselungsalgorithmen nicht klar getrennt

**Firewalls**

* zum Schützen aller möglichen Übergänge zwischen Netzwerksegmenten, Einschränkung der versuchten zur tatsächlich erforderlichen Kommunikation
* TCP/UDP
  + Filterung nach Empfänger- und Absenderadresse, Quell- und Zielport
  + Je nach Kommunikation weiter Filterkriterien wie Benutzer, HTTP-Header, Mailempfänger

Arten

* Paketfilter
  + zustandslos
  + Filterung der Datenpakete nach Sender/Empfänger-IP, Ports, Protokollen, Paketgröße
  + Probleme:
    - filtert einzelne Pakete, keine zusammenhängeden Ströme
    - keine Zustandsinformationen (Callback-Problem)
* Stateful Inspection
  + zustandsbasierter Paketfilter, Filterentscheidung abhängig von Paketverkehr
  + Filterung von UDP (Überwachung ausgehender UDP-Verbindungen, Protokollieren des spezifizierten Ports, Antwort nur an diesen Port akzeptieren)
  + Filterung nach Kontext (Paket ist Antwort auf Anfrage, ...)
  + Abwehr von DoS-Angriffen (Erkennen von SYN-Flooding, ...)
* Filtering Proxy
  + Proxy prüft die Zulässigkeit des Verbindungsaufbaus, legt Zustandsinformationen ab und verwaltet diese, über Filterung hinausgehende Sicherheitsdienste (z.B. Logging)
  + Application-Level-Gateway = anwendungsspezifischer Proxy-Dienst
    - zugeschnitten auf spezifische Protokolle (FTP, SMTP, ...)
    - Filterregeln erfordern Kenntnisse über Protokoll, spezifische Regeln
    - Vorteile
      * differenzierte Authentifikationen, feingranulare Kontrollen
      * Dienste können mit eingeschränkter Funktionalität betrieben werden
    - Problem: in der Regel nicht transparent, ursprüngliche Client-Server Verbindung ersetzt durch 2 Verbindungen
      * End-to-End Sicherheit unterbrochen, Man-in-the-Middle Attacken möglich
      * Zeitverzögerung durch Vermittlung

Hybride Firewalls

* Stand der Technik
* versuchen das Beste aus allen Welten zu bieten
* Kombination von
  + Stateful Inspection
  + Application-Gateways und Proxies
  + zusätzliche Funktionalitäten
    - VPN-Einwahl
    - Content-Filterung
    - Authentifizierung
    - Virenscanner ...
* Risiko: Zu viele Funktionalitäten erhöhen Fehlerrate, vergrößern Angriffsmöglichkeiten der Firewall

Personal Firewalls

* auf eigenem Computer installiert, schützen vor Angriffen
  + von außen: DoS-Angriffe, Einbruchsversuche
  + von innen: offene Ports, Abschottung lokaler Schädlinge (Viren)
* Vorteile:
  + skalieren, anwendungsnah, Ausbreitungsbekämpfung von Viren an der Quelle
  + bietet Grundschutz
* Nachteile:
  + Unkenntnis der Nutzer
  + ohne klar definierte Security-Policies untauglich (schnelle Bestätigung bei Verbindungserlaubnis)
  + unsicher durch Fehler im OS, Bedienungsfehler, Viren die Firewall aushebeln, ...

Policy-Designs

* stealth Regel: Zugriff auf Firewall ausschließlich zu Administrationszwecken
* default deny: verbietet alles, was nicht vorher erlaubt wurde

Change-Management

* Firewall wird von selbst mit der Zeit unsicher
* Betriebsprozesse müssen immer ein Teil des Firewalldesigns selbst sein

