Prinzipien der IT-Sicherheit

Prof. Dr. Hagen Lauer

IT-Trends

- **Simulation/Berechnung** — 1950 - heute — Atomwaffen, Rechnungswesen, KI, VR, Spiele
- **Vernetzung & Speicher** — 1980 - heute — Email, Messaging, Suchmaschinen, Social Media, Unterhaltung, Arbeit
- **Interaktion** — 2010 - heute — Cyber-Physische Systeme, IoT, IIoT, Smart Grid, Health IoT, autonomes Fahren
- **Integration** — 2040? — Integration von Computern und biologischen Prozessen (e.g., Bewegung, Wahrnehmung, Denken, …)

Trends

IT-Trends – Fazit

- Anwendungen vermehrt verteilt, vernetzt, dynamisch und heterogen
- Angriffe haben oft finanzielle Motive
- Hohe Anforderungen an Sicherheit und Zuverlässigkeit
- IT-Sicherheit und Datenschutz muss von Beginn an berücksichtigt werden!

  —> „Security by Design“ und Privacy by Design“

Trending: Sharing vs. Isolation

- Sicherheit durch physische Trennung von Ressourcen
  - **Leicht:** Bringe Daten, benutze Computer, nehme Daten wieder mit.
  - Container oder VM's könnten so etwas noch heute realisieren.

- Timesharing
  - **Schwer:** Jeder Nutzer will gefühlt „eigene Maschine", jedoch sollen *manche* Daten, Ressourcen und Programme geteilt sein.
  - Hier siegt oft die Bequemlichkeit, denn echte Isolation ist unbequem.

- Heute: Smartphones, Apps, Cloud-Computing, Cloud-Storage
  - Viele heterogene Apps (i.e., Social Media, Messaging, Banking, Arbeit)
  - Viele wertvolle Daten (Persönliche Daten, IP, Crypto, …)
- **Lösung:** Entwicklung *sicherer* Betriebssysteme?

Warum ist Sicherheit wichtig?

- Sicherheit ist wichtig für:
  - Betriebssicherheit (physischer Systeme)
  - Geheimhaltung (und Privatsphäre)
  - Funktionalität
  - Schutz von Werten (Information, Daten, Anlagen, Geräte, Systeme)
  - Unternehmen (IP, Kunden, Produkte, Wahrnehmung)
  - Stabilität von Staaten (KRITIS-relevante Sektoren)
  - …

Sicherheit im Sinne von Safety

- Safety: **Funktionssicherheit**
  - **System** verhält sich konform einer Spezifikation und erreicht definiertes Sicherheitslevel
  - **Safety** bezieht sich auf Vermeidung von potenziell gefährlichen Situationen oder Ereignissen
  - **Ziel** ist, die physische Integrität und Gesundheit von Menschen sowie die Umwelt und das Eigentum zu schützen […]
  - **Anwendung** in der Industrie, im Baugewerbe, Gesundheitswesen, Transportwesens und der Luftfahrt […]

Safety Fails

- Beispiele für Safety- Verletzungen
  - Therac-25: In den 1980er Jahren kam es bei der Therac-25, einem Strahlentherapiegerät, zu einer Fehlfunktion, bei der Patienten tödliche Strahlendosen verabreicht wurden. Ursache dafür war ein Programmierfehler, der es dem Gerät ermöglichte, ohne angemessene Sicherheitsvorkehrungen im "Hochleistungsmodus" zu arbeiten.
  - 1998: Mars Climate Orbiter Sonde zerschellte am 11. Dezember 1998 auf der Marsoberfläche, da sie 170 km tiefer als geplant flog. An der Mission waren zwei Teams der NASA beteiligt, eins rechnete in Metern, das andere in Fuß.
  - The Y2K Bug: Um die Jahrtausendwende gab es weit verbreitete Befürchtungen, dass Computersysteme aufgrund des "Jahr-2000-Fehlers" ausfallen würden. Ursache dafür war eine Programmierpraxis, bei der nur die letzten beiden Ziffern der Jahreszahl verwendet wurden, um Speicherplatz zu sparen, was zu der Befürchtung führte, dass die Systeme ausfallen würden, wenn die Jahreszahl von "99" auf "00" wechselte.
  - The Ariane 5 Rocket Explosion: In 1996 explodierte die Ariane 5 wenige Sekunden nach Start. Grund war ein Software-Fehler im Lenksystem, der zur Selbstzerstörung der Rakete führte.

Informationssicherheit (Security)

- Manchmal auch Synonym mit „Cybersicherheit“
- Umfasst u.a. IT-Sicherheit, Computersicherheit, Datensicherheit und Datenschutz, IoT-Sicherheit, Cloud-Sicherheit, …

- **Ziele**:
  - Schutz von Informationen vor unbefugtem Zugriff, Diebstahl, Verlust oder Beschädigung.
  - Informationen sind vertraulich, integritätsgeschützt und verfügbar.

- **Umfasst technische** und **organisatorische Maßnahmen** wie
  - Verschlüsselung und Zugriffskontrollen,
  - Schulungen und Sensibilisierung der Mitarbeiter.
  - Informationssicherheit betrifft alle Arten von Informationen, einschließlich persönlicher Daten, Geschäftsgeheimnisse und geistiges Eigentum.

„Bugs“ und die Informationssicherheit

- Heartbleed-Bug (2014):
  - ein Fehler in der kryptografischen Bibliothek OpenSSL
  - Zugriff auf sensible Daten, einschließlich Passwörter und private Schlüssel, von Millionen von Websites.
  - Der Fehler blieb mehrere Jahre lang unentdeckt

- Meltdown und Spectre:
  - 2018 entdeckt (u.a. Graz)
  - zwei große Sicherheitslücken in modernen Computerprozessoren
  - Angreifern greifen auf sensible Daten im Speicher zu.
  - Die Fehler betrafen fast alle modernen Prozessoren und erforderten Software-Patches zur Behebung

SPECTRE

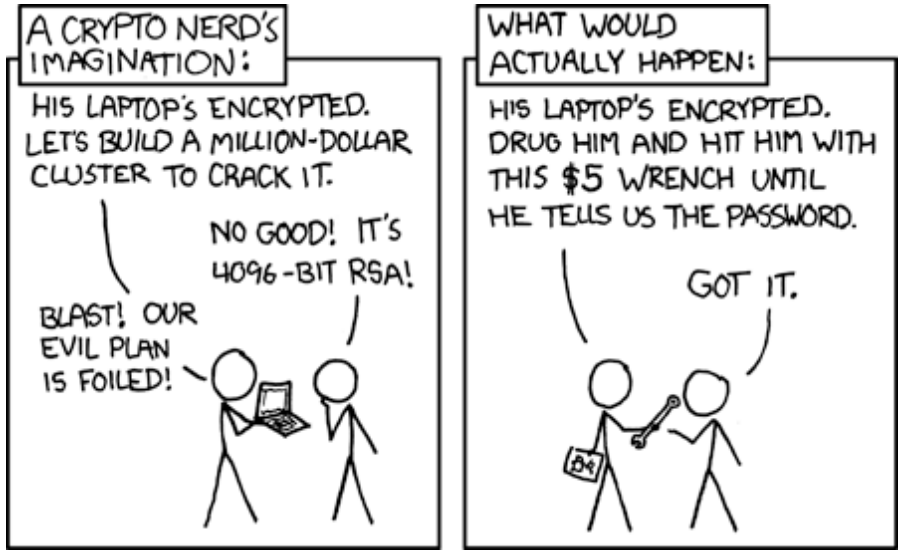
...

- SolarWinds Supply Chain Attack (2020):
  - Groß angelegter Cyberangriff auf SolarWinds
  - SolarWids bietet IT-Verwaltungstools für viele Regierungsbehörden und Privatunternehmen an
  - Angriff, der vermutlich von russischen Hackern, beeinträchtigte die Software von SolarWinds und ermöglichte den Hackern den
  - Zugriff auf sensible Daten mehrerer Organisationen.
  - Wichtiger „Weckruf“ für die Sicherheit der Lieferkette

Sicherheit in der echten Welt

Dreht sich um **Wert**, **Hürden** und **Abschreckung**.

- Wert (und Risiken)
  - Management: (*Schutz*) < (*Schadensfall*)
    - “Perfect Security” ist teuer, unerreichbar
  - Angriff: (*Gewinn*) > (*Kosten*) + (*Konsequenzen*) * (*Wahrscheinlichkeit*)
- Hürden
  - Schlösser, Tresore, Mauern, Türen, …
- Abschreckung
  - Soziale Strukturen, gesellschaftliche Normen, Gesetze, Strafverfolgung



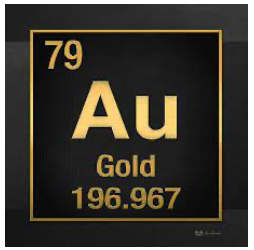
„*Here is a challenge: Computers are as secure as real world systems, and people believe it*“.—Butler W. Lampson

Schutzziele — Ziele der Informationssicherheit

Sicherstellung gewünschter Schutzziele *unter widrigen Umständen*:

Vertraulichkeit (**c**onfidentiality)
Integrität (**i**ntegrity)
Verfügbarkeit (**a**vailability)

Authentifizierung (authentication)
Autorisierung (authorisation)
Auditing (Prüfbarkeit)



Verbindlichkeit (Nicht-Abstreitbarkeit, non-repudiation)
Privatheit (Privacy, Anonymity)
…

Grundprinzipien der IT-Sicherheit

Kenne die Bedrohungen

- Schutzgut *(Ziel)*

- Gefahr *(kein Zusammenhang)*

- Bedrohung *(Gefahr mit Bezug zum Ziel)*

+

- Schwachstelle *(Mangelnde Schutzmaßnahmen)*

- Angriff *(Nutzt Schwachstelle aus)*

Kenne die Bedrohungen

- Katastrophen Warnsystem („Bundeswarntag“ 2020)
  - Ziele: **Verfügbarkeit** (Availability), **Integrität** (Integrity), **Authentizität?** (Authenticity/Authentication)
- Smart Grid / Teslas virtuelles Kraftwerk (2021)
  - Ziele: **Verfügbarkeit**, **Integrität**, **Geheimhaltung/Privatheit**, …
- Online Banking / Brokering (2021)
  - Ziele: **Authentizität, Geheimhaltung**, **Autorisierung**, (**Integrität**, **Verfügbarkeit)**

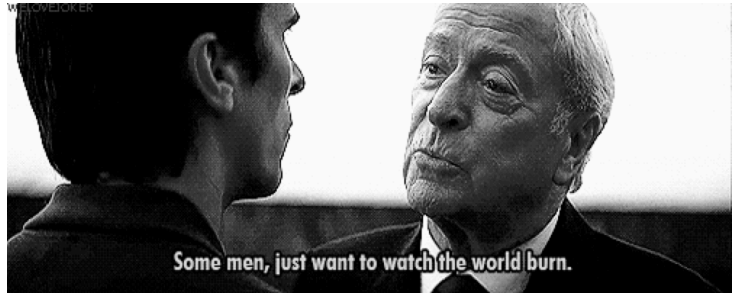
Kenne die Bedrohungen

- **Attacker**
  - System, Person oder Personengruppe, die Angriffe durchführen
  - Beispiel: Hackergruppen, Saboteure

- **Attackermodel**
  - Zur Erfassung der verschiedenen Gefährdungslagen zur Risikoabschätzung
  - Beschreibt u.a. Angreifertyp (z.B.: Skript Kiddie, Hacker, Mitarbeitende, Wirtschaftsspione, Regierungsbehörde), Fähigkeiten (Insider / Outsider, Werkzeuge, kryptografische Fähigkeiten), Budget und Ziele einer Angreiferin

Geld

Politik



Spaß?

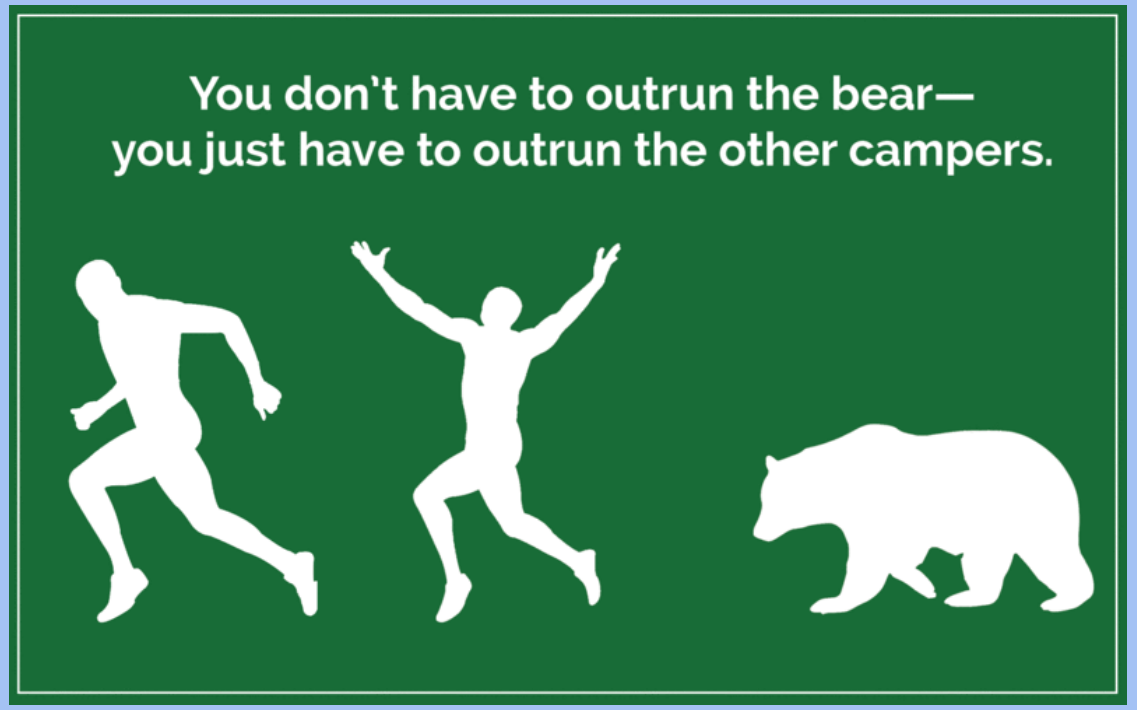
Kenne die Bedrohungen

- **Hacker**
  - White-Hat: Gesetze, Hackerethik zum Aufdecken von Sicherheitslücken
  - Grey-Hat: Verstoßen teilweise gegen Gesetze / Hackerethik
    jedoch nur, um ein höheres Ziel zu erreichen
  - Black-Hat: Handeln mit krimineller Energie

- **Motivation** der Angreifer
  - Geheimdienste: Spionage, Sabotage und Überwachung
  - Unternehmen: Wirtschaftsspionage (Zusammenarbeit mit Geheimdiensten)
  - Professionelle Schattenwirtschaft (Auftragshacking): monetäre Gründe z.B.
    - Fälschung von PayTV-Karten
    - Abgreifen von Kreditkarteninformationen (Warenkreditbetrug)
    - Phishing-Angriffe im Bereich Online-Banking
    - Ransomware
  - Whistleblower: Veröffentlichung geheimer Informationen
  - Politisch motivierte Angriffe

Grundprinzip: Kenne die Bedrohungen

- Kenne dein Bedrohungsmodell
  - Kenne das Schutzgut
  - Kenne deine Schutzziele
  - Kenne Gefahren
  - Kenne Angreifer



Sicherheit ist Ökonomie

- **Risiko = Eintrittswahrscheinlichkeit * potentieller Schaden**
  - Risiko dient der Priorisierung umzusetzender Maßnahmen
  - Risikobewertung erfolgt anhand des definierten Angreifermodells

- **Eintrittswahrscheinlichkeit**
  - Abschätzung von Aufwand, Motivation und Nutzen für den Angreifer

- **Schadenhöhe**
  - Abschätzung der Schadensfälle und Folgen erfolgreicher Angriffe
    - **Primäre Schäden**, z.B. Produktivitätsausfall, Wiederbeschaffungs- oder Wiederherstellungskosten
    - **Sekundäre Schäden**, z.B. Imageverlust

MATRIX ZUR RISIKOBEWERTUNG

Wahrscheinlichkeit	Auswirkung/Schaden			
	Niedrig	**Mittel**	**Hoch**	**Sehr hoch**
Sehr wahrscheinlich	gering	mittel	hoch	**sehr hoch**
Wahrscheinlich	gering	mittel	hoch	hoch
Möglich	gering	gering	mittel	mittel
Unwahrscheinlich	gering	gering	gering	gering

BSI. IT-Grundschutz, Risiken bewerten

Sicherheit ist Ökonomie

- Kosten/Nutzen-Analyse:
  - Die Kosten der Abwehr sollten geringer sein als der erwartete Schaden
  - Sicherheit ist teuer und kann beliebig teurer werden. Teuer bedeutet leider oft besser.
  - Wenn der Angriff teuerer ist als die Beute, dann sind Angriffe unwahrscheinlich

- Beispiel: 10€ Schloss für 1€ Wertgegenstand…
  - … außer der 1€ Gegenstand kann wiederum für einen Angriff verwendet werden

- Beispiel: Sie entdecken eine neue 0-Day-Lücke, die überall funktioniert. Diesen Angriff würde man nicht verschwenden, um eine beliebige Person an zu greifen.
  - iPhone 0-Days kosten schnell ~$1M und werden gehortet.

Grundprinzip: Sicherheit ist Ökonomie

- Sicherheit lässt sich immer erhöhen und verbessern
- Teurer ist oft auch besser
- Muss zum Schutzgut passen
- Muss den „zu erwartenden“ Angriffen standhalten
- Spezifikationen sind hilfreich, um „Grundschutz“ her zu stellen

Schraubendreher
5 Minuten

Stahlbohrer
(mehrere),
30 Minuten

Oceans Eleven,
das A-Team
und James Bond,
Mindestens 120 Minuten
Spielfilm

Keine „Security Through Obscurity“

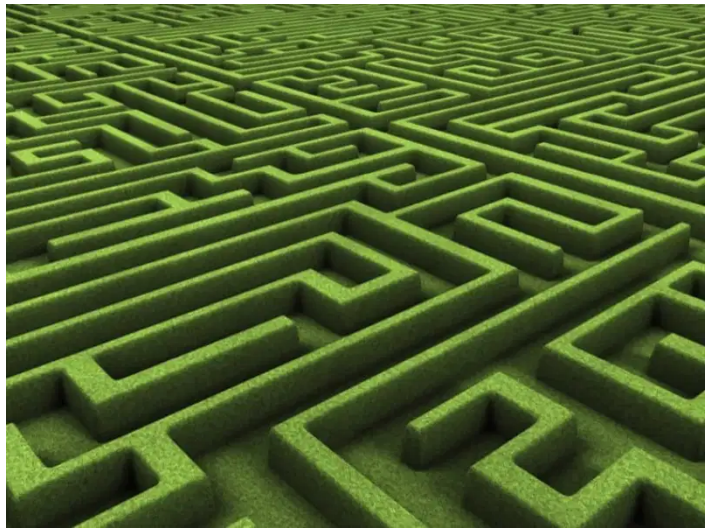
- Kerckhoffs Prinzipien für Krypto:
  - …
  - **It must not be required to be secret, and must be able to fall into the hands of the enemy without inconvenience**
  - …

Angreifende dürfen Verfahren kennen, nicht den Schlüssel.

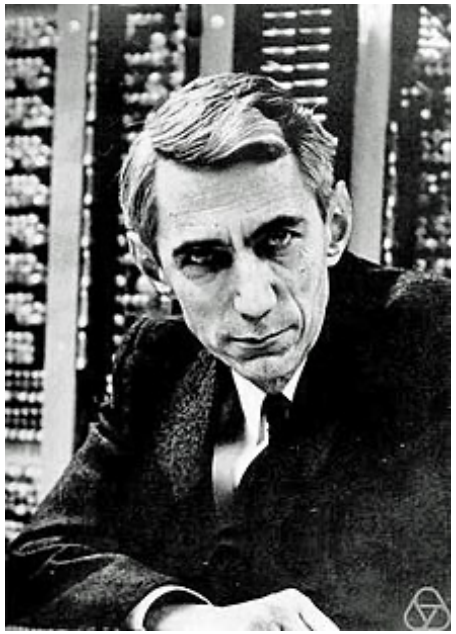
Auguste Kerckhoffs
(1835-1903)

Keine „Security Through Obscurity“

- Shannons Maxime:
  - „The enemy knows the system.“
  - Erweiterung von Kerckhoffs Prinzip



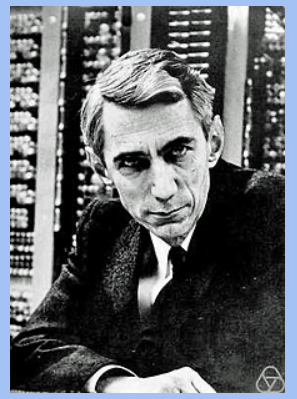
Beispiel Labyrinth: Ist der Weg einmal bekannt, dann ist das System „gebrochen“.



Claude Shannon
(1916-2001)

Grundprinzip: Keine „Security Through Obscurity“

- Oft verwirrend — für beide Seiten
- Typischerweise komplex im Entwurf und dementsprechend auch teuer
- Durch diese Eigenschaften sind solche Maßnahmen selbst of fehleranfällig
- Wird generell als ineffektiv und riskant betrachtet — warum der Einsatz?


Claude Shannon
(1916-2001)


Auguste Kerckhoffs
(1835-1903)

„The enemy knows the system“.—Claude Shannon

Security By Design

- Sicherheit von Beginn an, als fester Teil des Systementwurfes
- Schreibe eine Spezifikation, oder:
  - „*Wie man ein System versteht bevor (oder nachdem) man es gebaut hat.*“—B. Lampson
  - Kein generelles Modell oder Axiome für IT-Sicherheit
  - Unterscheidung: **Präzise** vs. **ungenaue** Systeme
    - **Präzise**: Banken, SCADA, IIoT, Flugsteuerung …
    - **Ungenau**: Google Suche, News, Social Media
- Zurück zur Planung: Welche Sorte System liegt vor?

Erstelle Modell -> definiere Aktionen -> überprüfe ob „Code“ abstrakte Aktionen implementiert - einfach, oder?

Security By Design

- „Formale Verifikation“
  - Fast ausschließlich in der Theorie (Dijkstra, Hoare) in den 70ern (für 0-100 LOC)
  - Heute skaliert Verifikation viel besser:
    - CompCert: Verifizierter C-Compiler
    - seL4: Verifiziertes Microkernel OS
    - Viele weitere, inklusive sog. *sound* Programmiersprachen
  - Viele nützliche Helferlein:
    - Mengen, Funktionen, Relationen, Graphen
    - Zustandsautomaten (Automatentheorie allgemein)
    - Software:
      - Coq, Caml, OCaml + libs, Haskell-to-*
      - Isabelle/HOL + Cake
      - MSR’s Lean Theorem Prover, Z3, F*(star)

„Logic, like whiskey, loses its beneficial effect when taken in too large quantities.“ — Lord Dunsany

Security By Design

- **Menschlich**
  - Code Reviews
  - Extreme/Pair Programming
- **Methodisch**
  - Design Patterns
  - Test-driven Development
  - Version Control
  - Bug Tracking
  - DevSecOps
- **Technologisch**
  - Statische Analysen
  - Fuzzing
- **Mathematisch**
  - Strenge Typisierung
  - „Formale“ Verifikation

Weniger formal: Probleme im Design / Programm können „übersehen“ werden.

Alle Methoden sollten eingesetzt werden!

Auch formale Methoden haben Lücken:
- Habe ich das „Richtige“ bewiesen?
- Stimmen die Annahmen?

Formal: Eliminiere *mit Gewissheit* so viele Probleme wie möglich.

„Logic, like whiskey, loses its beneficial effect when taken in too large quantities.“— Lord Dunsany

Grundprinzip: Security By Design

- Beachte Planung!
  - Assets? Schutzziele?
  - Präzise / weniger präzises System?
- Schreibe eine Spezifikation
  - Modelliere das System, Aktionen
  - „Simuliert“ Code das Modell/Aktionen?
- Softwareentwicklungsprozess beachten
  - Über 50 Jahre Erfahrung in der Entwicklung werden nicht durch ein „Tool“ ersetzt
  - Programme werden von Menschen entworfen und geschrieben

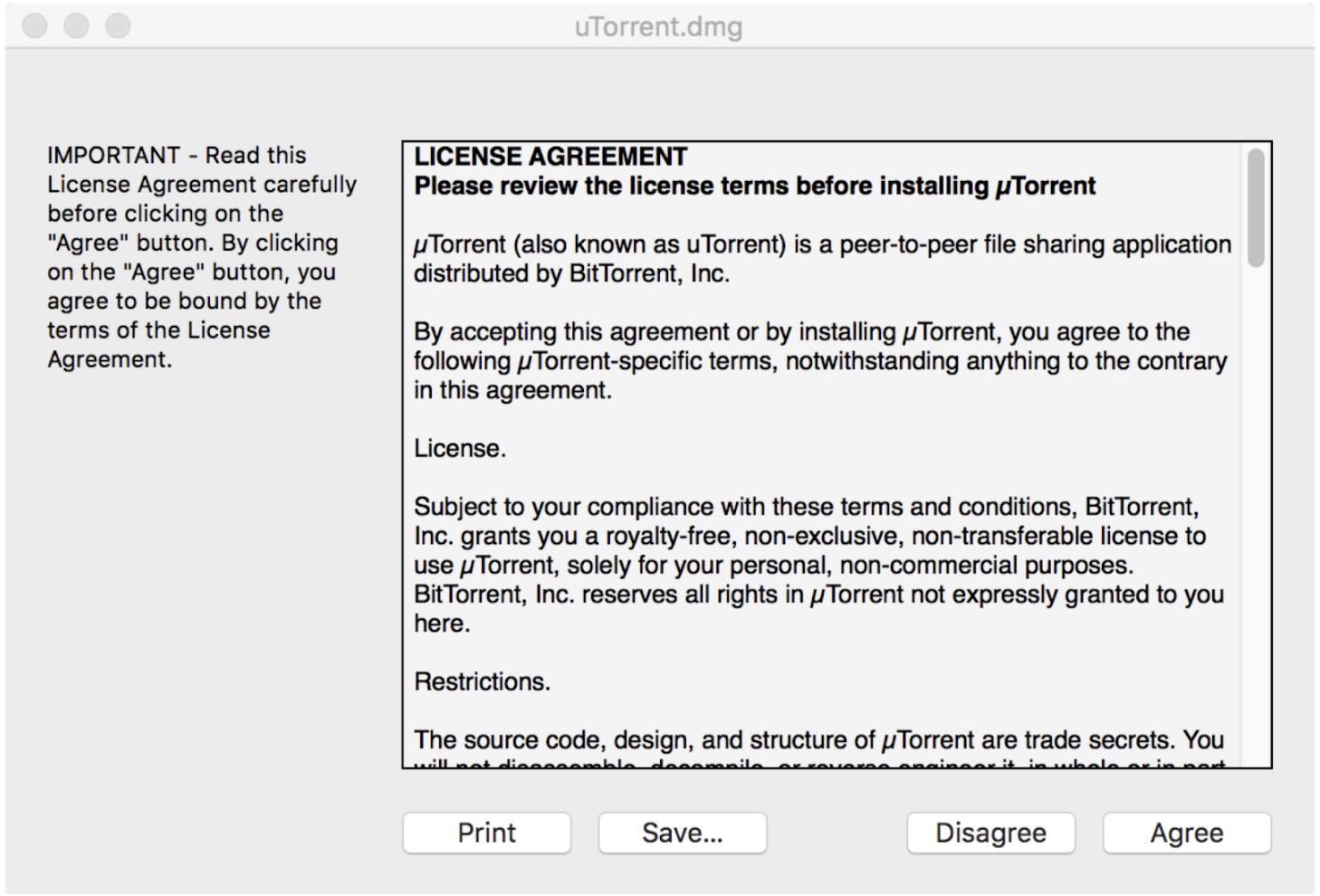
„A point of view is worth 80 points of IQ.“—Alan Kay

Least Privilege

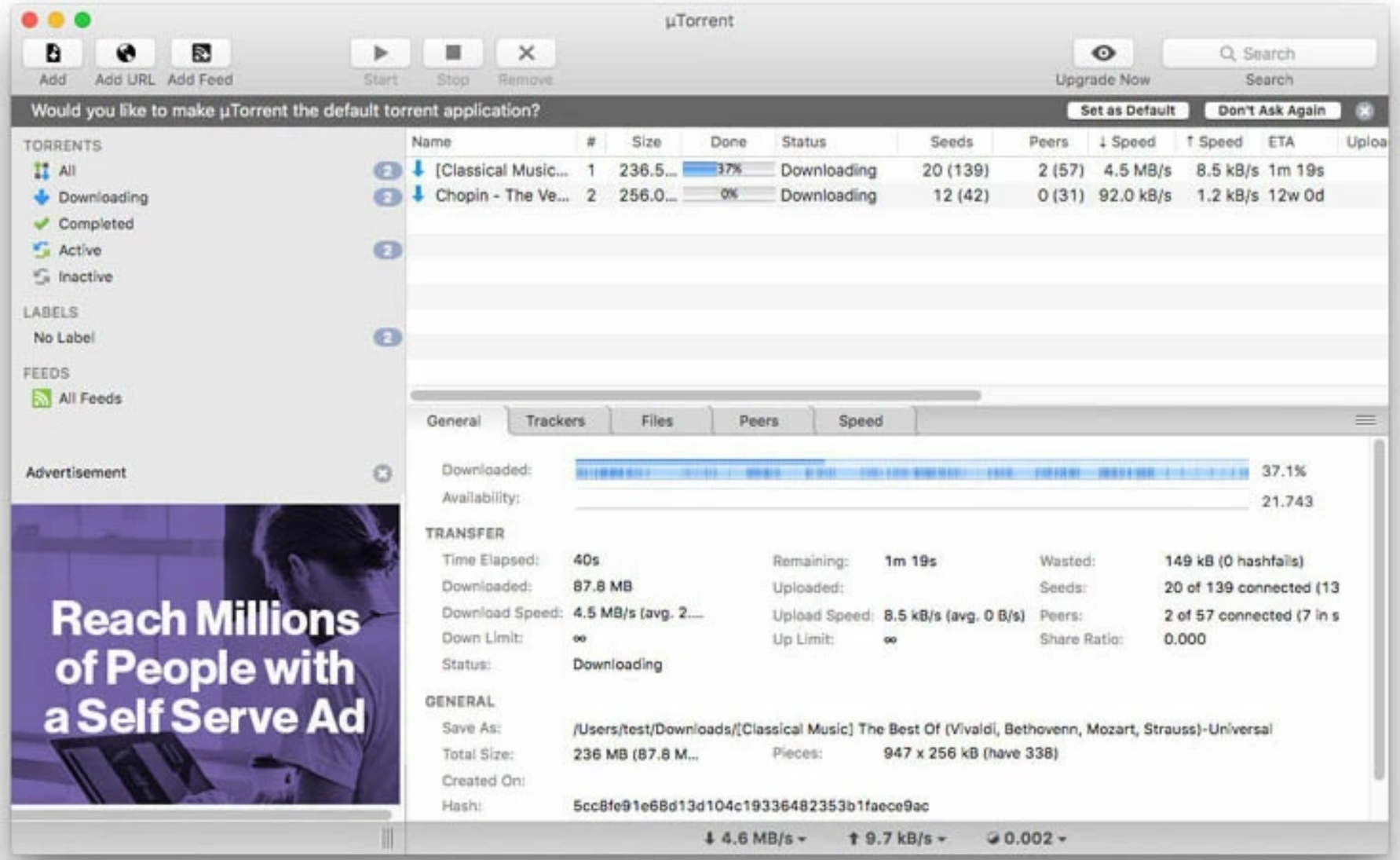
- Optimal: Ein Programm bekommt nur das *Minimum an Rechten, die es braucht*, um korrekt zu funktionieren.
  - Zu viele Berechtigungen führen oft zu sog. „Leaks“ und Missbrauch durch Dritte.

- Die Basis dafür stellt oft das Betriebssystem dar.
  - Isolation + Zugriffskontrolle
    - **Isolation**: Prozesse können nicht auf fremden Speicher zugreifen
    - **Zugriffskontrolle**: Prozesse kann nur auf Dateien mit der nötigen Berechtigung zugreifen
  - App-Entwickler/innen tragen Verantwortung - kaum jemand *studiert* Berechtigungen.

Least Privilege



Least Privilege

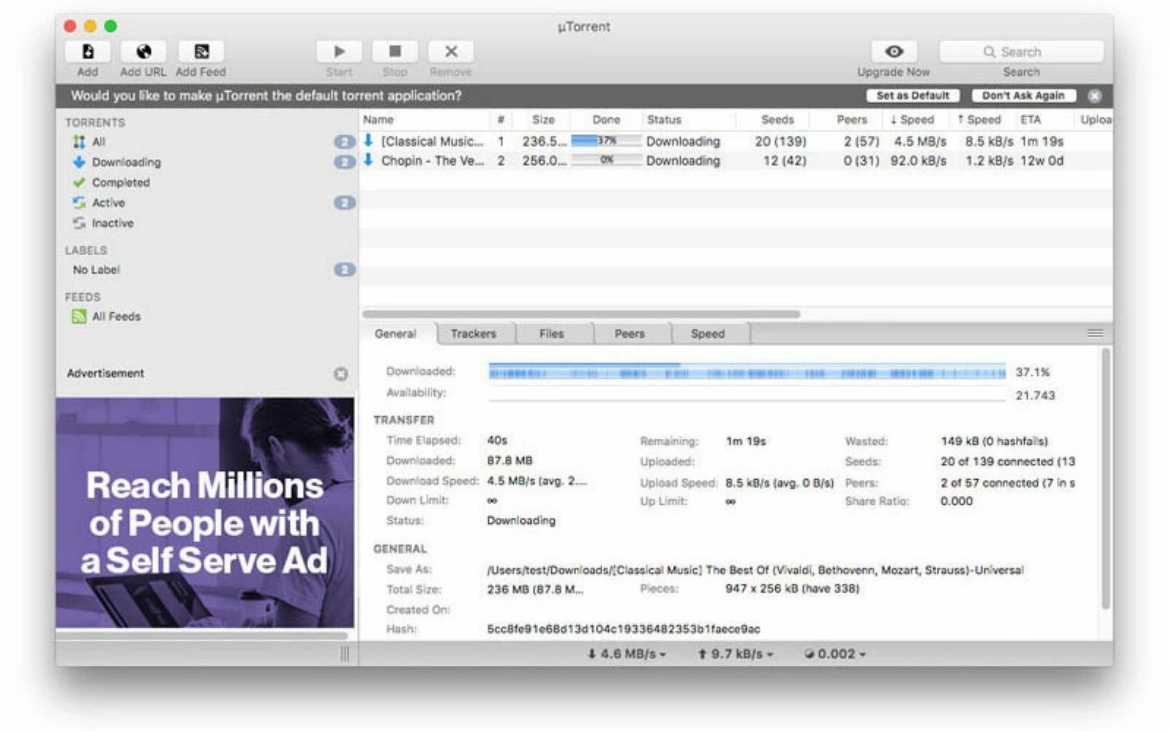


Grundprinzip: Least Privilege

- Was kann so ein Programm:
  - Dateien preisgeben („leaken“), schreiben, löschen
  - Spam versenden (bidirektional)
  - Code ausführen

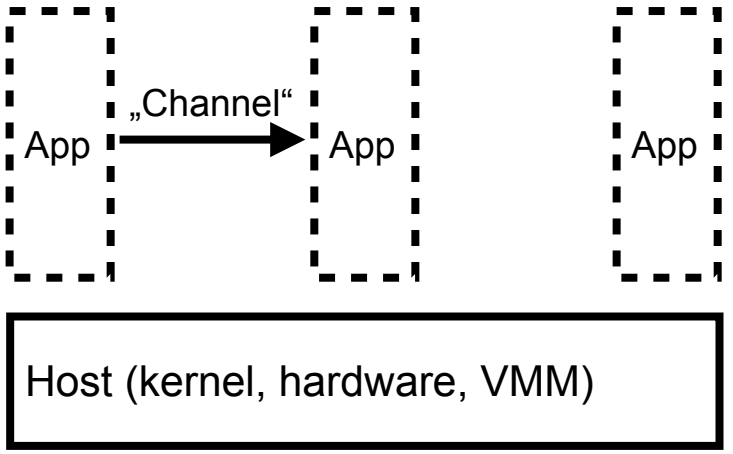
- Was <u>muss</u> das Programm können?
  - Bildschirmanzeige (Fenster öffnen)
  - Berechtigung für *manche* und <u>nicht alle</u> Dateien
  - Internet Verbindungen (beschränkt)

- **Berechtigungen einschränken.**



Zugriffskontrolle

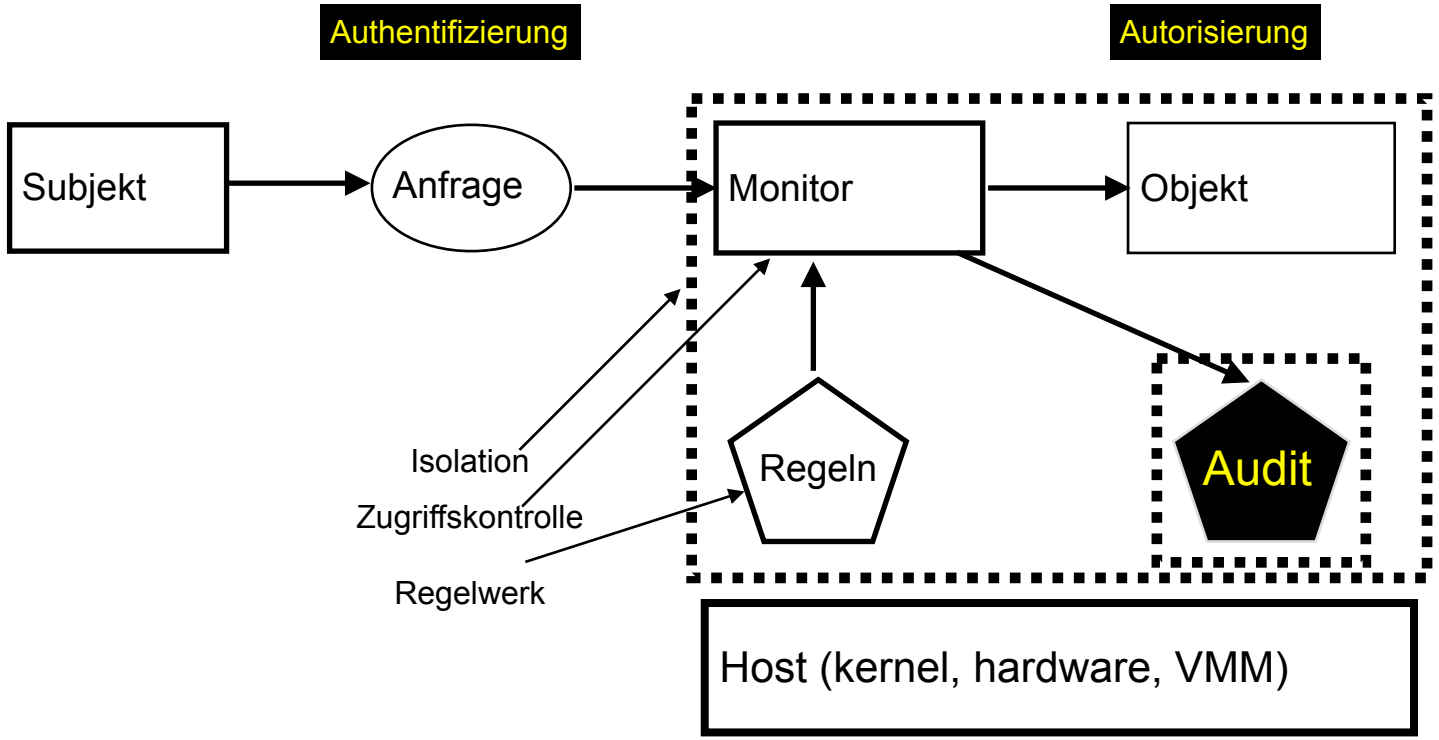
- Grundlage: **Isolation**
  - Host isoliert Laufzeitumgebung (App)
  - Basis für jegliche IT-Sicherheit
  - Host muss *vertrauenswürdig* sein.

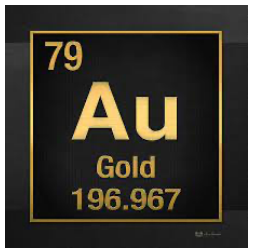


- Vielzahl von Mechanismen mit unterschiedlicher Robustheit:
  - Java und JVM
  - Module o. Objekte in einer Sprache / Laufzeitumgebung
  - Prozesse in Betriebssystemen
  - Virtuelle Maschinen, Container im Falle von Virtualisierung (e.g., KVM, Docker)
  - Krypto in Netzwerken

Zugriffskontrolle

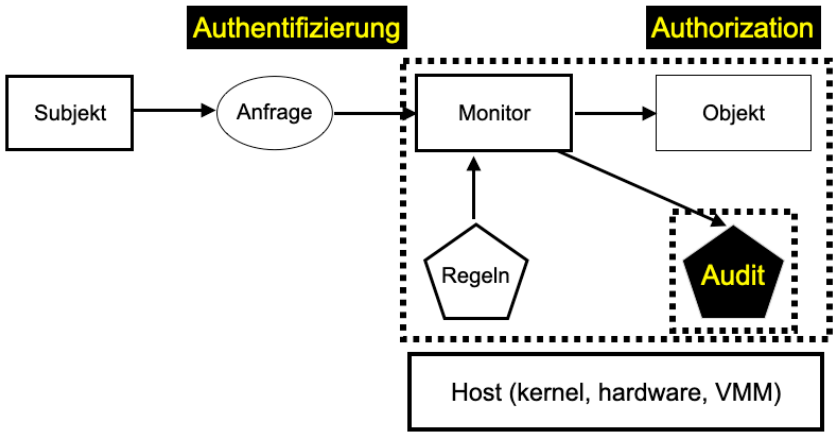
1. Isolationszonen / Grenzen schränkt Attacken auf Kanäle ein
2. Zugriffskontrolle für Datenfluss
3. Regelwerk, wer darf was?





Zugriffskontrolle

- **Authentifizieren**: Subjekte, *wer* stellt *Anfrage*?
  - Leute, Kanäle, Prozesse, Server, Programme…
- **Autorisieren**: *Wem* darf eine *Ressource* anvertraut werden?
  - Gruppen, Subjekte, oder Ressourcen
    - Definierbar durch Rollen, Fähigkeiten, Eigenschaften
- **Audit**: *Wer* hat *wann was* gemacht?



Zugriffskontrolle

- Guard berechnet: *permissions* = *regel*(*subjekt, objekt)*
  - Wenn das zu mathematisch klingt … nennt es Matrix:

Subjekt	***Objekte***
	Datei XYZ
Alice	read, write
Bob	read
Malory	—

- Entscheidend: Auswahl des Mechanismus DAC, MAC, RBAC?
  - **Discretionary AC**: Wer darf was mit dem **Objekt** anstellen? **EigentümerIn** entscheidet.
    - Ist EigentümerIn dazu wirklich in der Lage?
  - **Mandatory AC**: Wer darf **generell** was? **Admin** entscheidet.
    - Ist Admin vertrauenswürdig? Sagt das System nicht nur „Nein“?
  - **Role-based AC**: Welche Gruppe/Rolle darf was? **EntwickerIn** entscheidet.
    - Admins weisen nun *lediglich* User Rollen zu. Das Regelwerk ist jedoch besiegelt.

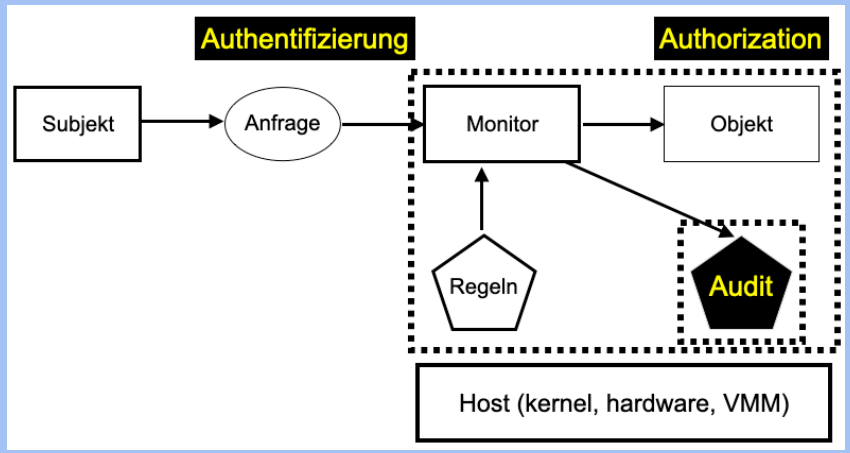
Grundprinzip: Zugriffskontrolle

- Isolation ist Grundlage

- Gute Zugriffskontrolle ist „Gold-Standard“ für Daten- und Informationssicherheit
  - *Kryptografie ersetzt diese nicht, sie unterstützt.*

- Definition von Regelwerk (Policy) u.U. sehr komplex und muss auf Applikation angepasst werden

- Mechanismus entscheidet über Funktion im Betrieb:
  - *Steht die Zugriffskontrolle scheinbar im Konflikt mit der Aufgabe, so wird sie umgangen.*



Faktor Mensch

- **Benutzer**
  - Mögen Bequemlichkeit
  - Mögen weder umständliche IT-Sicherheit noch IT-Sicherheitsbeauftragte
  - Finden eine Möglichkeit zur Umgehung, wenn es ihre Arbeit erleichtert
- **Programmierer**
  - Machen auch mal Fehler
  - Müssen manchmal Mittel (Tools) verwenden, die Fehler ermöglichen (e.g., C und C++)
- **Andere**
  - Sog. Social Engineering Attacken nutzen soziale Beziehungen und Vertrauen aus
- **IT-Produkt muss den „Gun-Foot“-Test bestehen:**
  - BenutzerInnen müssen möglichst davon bewahrt werden, sich selbst zu schaden (Fool-Proof).

Faktor Mensch

- **Social Engineering**:
  - Angriff funktioniert nur mit Menschen
  - Ziele sind typischerweise primitiv:
    - Lasse jemanden etwas glauben
    - Lasse jemanden auf etwas vertrauen
    - Bringe jemanden dazu, etwas zu erlauben
    - Bringe jemanden dazu, Zugang zu gewähren
    - Bringe jemanden dazu, etwas aus zu händigen (Information, Passwort, Schlüssel…)

- Solche Attacken involvieren fast immer Fälschungen (Identitäten, Adressen, Konten, Karten, Ausweise, …)
- Spam, Phishing, Spoofing, Junk-Mail

Grundprinzip: Der Faktor Mensch

- Produkte nie ohne „User-Tests“
- Unbedingt einen Entwicklungsprozess etablieren
  - *Audits sind enorm wichtig, um Fehler zu finden und Compliance zu dokumentieren*
  - *Audits sind Kernpunkt von MS SDL, DevSecOps*
- Gesamte Kette in der Entwicklung beachten
  - *Supply Chain, Mittel, ProgrammiererInnen, Admins, Kunden, Support, …*
- Menschen müssen geschult werden und offensichtliche Gefahren erkennen können

„Egal wie sicher ein Systementwurf ist, entscheidend ist der Mensch.“—Unbekannt

Defense in Depth

- Typischerweise werden mehrere Arten von Schutzmechanismen kombiniert, um Schutzziele zu erreichen und zu verstärken

- Angreifende müssen im optimalen Fall alle Schutzmechanismen überwinden, um ans Ziel zu gelangen
  - Idealerweise entstehen Synergieeffekte
  - Oftmals ist jedoch „Patchwork“

- Erinnerung: Sicherheit ist *auch* Ökonomie
  - Verteidigung kann beliebig viel Geld und Aufwand kosten ohne klaren Nutzen
    - 101 Firewalls sind nicht viel besser als 100…

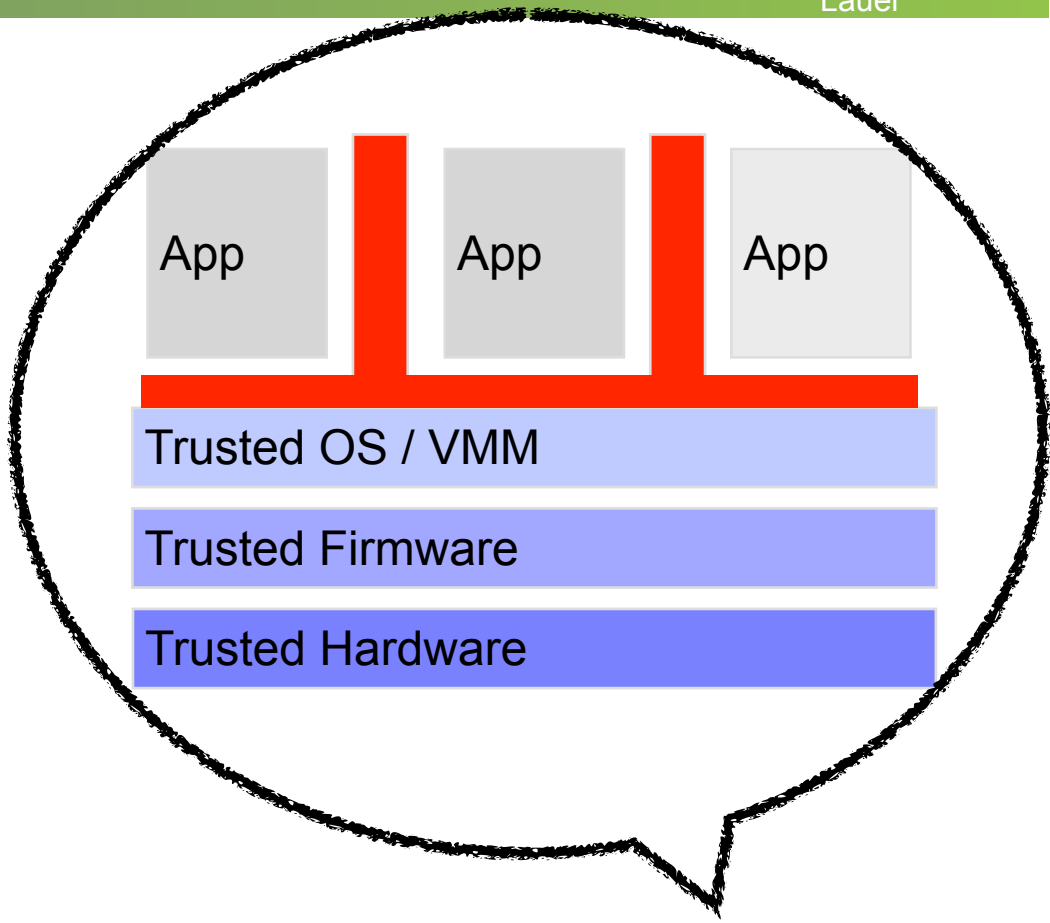
Defense in Depth

- **Patchwork**
  - Firewalls
    - *„Löcher von außen Stopfen“*
  - Signatur-Erkennung
    - *„Bekannte Angriffe vielleicht erkennen“*
  - Memory Safety
    - *„Verhindert grobe Programmierfehler“*
  - Intrusion Detection
    - *„Unbekannte Angriffe vielleicht erkennen“*
  - Control Flow Integrity
    - *„Blockieren von „Jumps“, die nicht erwartet werden“*
  - Security Prozeduren (e.g. Microsoft SDL)
    - *„Zwang als sicher erachtete Tools zu verwenden“*

Prinzipiell sind dies keine schlechten Lösungen, es sind aber eher Bandagen als Heilsbringer.

Defense in Depth

- **Trusted Computing Base**
  - Trusted Hardware (Logging, Identity, Key-Management)
  - Verified Firmware (Open Source, verifiziert, secure boot enabled)
  - Vertrauenswürdiges Betriebssystem (OS/VMM)
    - Tut genau das, was es soll (und nichts weiter)
    - In Verbindung: Korrekt, Vollständig, und Sicher
- Spec und Code stimmen, dann kann die Konfiguration immer noch falsch sein
  - Konfiguration ist immer individuell
- Jedoch widersprüchlich:
  - Komplex vs. Sicher („small TCB“)
  - Einfachheit vs. Funktionsreich



(1982-heute)

„The price of reliability is the pursuit of the utmost simplicity.“—C.A.R. Hoare

Grundprinzip: Defense in Depth

- Verteidigung auf verschiedenen Ebenen
  - Host (multiple Layer)
  - Netzwerk (multiple Layer)
- Schutzmaßnahmen müssen ineinander greifen
  - Hardware -> FW -> OS -> App
  - Prozeduren, Tools, Firewall, IDS, …

Festung Wilhelmstein

Keine Verteidigung ist jemals perfekt!

Monitoring trotz Schutzmaßnahmen

- „Vertrauen ist gut, Kontrolle ist besser.“
  - Taucht in unzähligen Sprachen so oder ähnlich auf…
  - Weisheit daraus: Vorbereitung, Schutz und Vertrauen scheitern und ersetzen kein Monitoring des Produktes
- Schutzmaßnahmen funktionieren oft nicht (mehr):
  - Zeit im Sinne der Angreifenden
    - Legacy Systeme, keine Patches, veraltete Krypto…
    - Warum gelingen Gefängnisausbrüche?
      - Design und Bau: 3 Jahre
      - Insassen: 15+ Jahre
    - Wie hoch war das Security-Budget? -> *Sicherheit ist Ökonomie*.

Monitoring und Response ist in jedem Fall unerlässlich.

Monitoring trotz Schutzmaßnahmen

- **Angriffserkennung**
  - Die Erkennung von Angriffen beruht auf Messungen des Systemverhaltens und Systemzuständen, die Indizien für mögliche Angriffe sind.
    - Boot-Logs, System-Logs, Network-Traffic, User-Login-Times, CPU-Activity, externe Faktoren…
  - Gewöhnlich werden dazu Regel-basierte Verfahren eingesetzt, die Zugriff auf Datenbanken für Angriffe haben:
    - Malware Definition Strings („Virusscanner“)
    - Email Headers und Sender-Listen (Spam-Filtering)
    - Packet- und Content-checking (Deep Inspection)
- **Anomalieerkennung**
  - Weniger spezifische CVEs, Listen, etc. stattdessen Indicators of Compromise und „gelerntes Normalverhalten“
    - **Vorteil**: Wesentlich bessere Erkennungsraten, Möglichkeit auch *Effekte* von 0-Days zu erkennen
    - KI-Basierte Erkennung ist erhöht Vorteil zwischen Kosten (Defense) und Kosten (Offense) deutlich
    - **Problem**: Anomalie ist nicht immer Angriff und nicht jeder Angriff resultiert in Anomalien

Grundprinzip: Monitoring trotz Schutzmaßnahmen

- Energiesysteme sind relativ heterogen und teilweise veraltet
- Schutzmaßnahmen sind nicht aktuell, nicht umsetzbar
- Angriffserkennung ist leider der effektivste Schutz
  - Ermöglicht effektive Gegenmaßnahmen:
    - Isolation
    - Separation
    - Wiederherstellung

Projekt SecDER, FhI SIT, Fh IEE (2021-2024)

„Vertrauen ist gut, Kontrolle ist besser.“

Grundprinzip: Vollständige Problembehandlung

- Fragestellungen:
  - Sind wirklich alle Schwachstellen behandelt?
    - Common Vulnerabilities and Exposures (CVEs)
    - Schnittstellen, Interfaces, offene Ports, Speicherorte, Wechselplatten, Sticks…
  - Gibt es Schnittstellen, durch die Daten und Informationen müssen?
    - Wenn ja, haben wir dort „Monitore“?
    - Können wir Ereignisse beobachten und aufzeichnen?
- Wächter/Monitore sind enorm wichtig:
  - Korrektheit
  - Vollständigkeit (kein Bypass möglich)
  - Sicherheit (sind selbst abgesichert…)
  - Sind eigentlich Teil der TCB

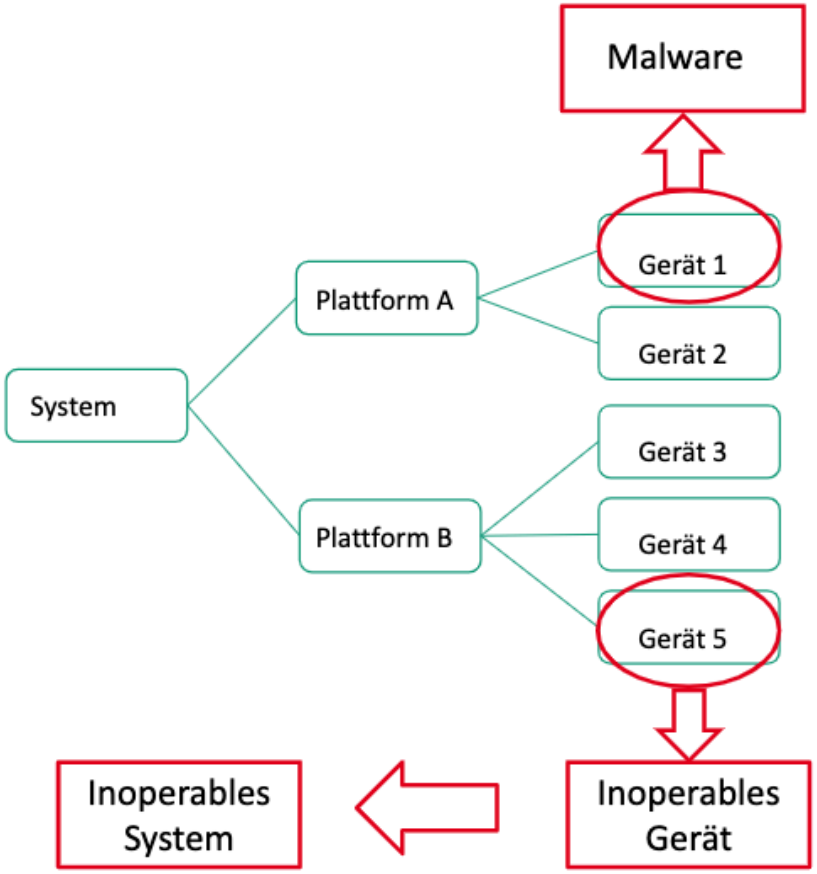
Fail-Safes und Wiederherstellung

- Fail-Safes sind existenziell für die Betriebssicherheit
  - Redundanz, manual Overwrite, Wiederherstellungsoption
  - Balanciert Security mit Safety

- **Idee**: Wenn Angriffe oder Ausfälle erkannt sind, dann fällt das System in einen „Notlaufmodus“ zurück
  - Notlaufmodi sind minimalistisch, besonders abgesichert, und zur Überbrückung gedacht
  - Systeme können dabei von innen und aussen isoliert werden (Ausschluss aus vertrauenswürdigen Systemen)
  - Fehlerisolation und Schadenbegrenzung

Fail-Safes und Wiederherstellung

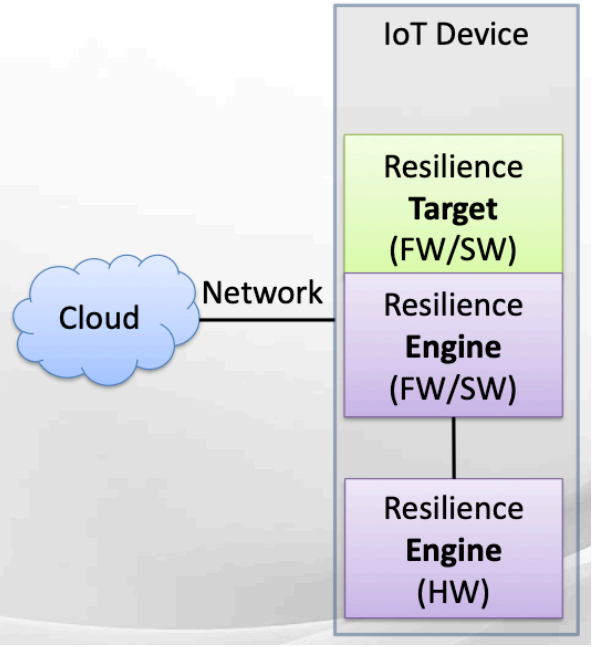
- Warum Fail-Safes und Wiederherstellung?
  - Systeme bestehen aus Plattformen
  - Plattformen wiederum aus Geräten
  - Einzelne Geräte sind wichtig für die **Verfügbarkeit** und **Integrität** des Systems
    - Angriffe auf Geräte bedrohen das System!
  - Ausfall von Geräten kann zum Versagen des Systems führen
    - Angriffe können so permanente, physische Schäden verursachen
- Beispiel: Netzbetriebsführung unseres „intelligenten“ Stromnetzes



Fail-Safes und Wiederherstellung

- Wiederherstellung: Wie?
  - Bsp: „*Totmannschalter*“
    - Fällt das Ziel aus oder ist es kompromittiert, so kann es diesen Zustand nicht beibehalten oder eine Wiederherstellung wird ausgelöst

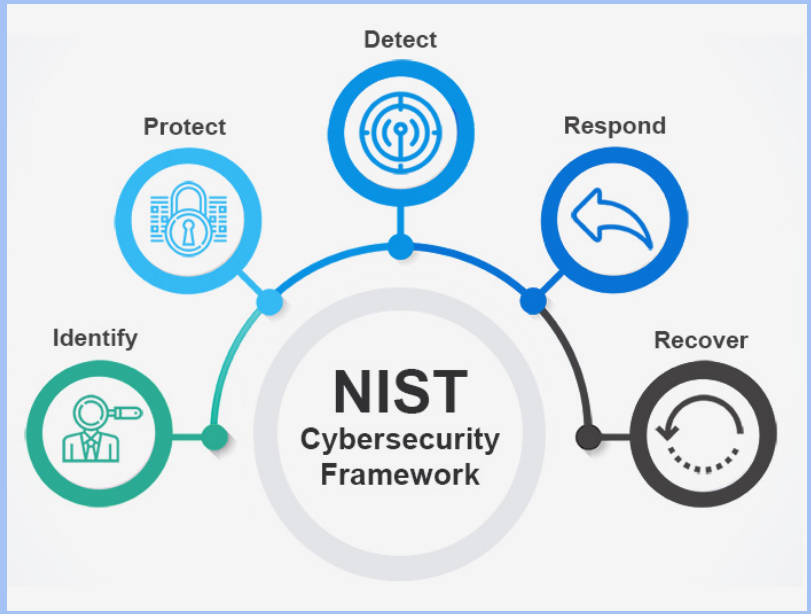
- **Vorteil**: Tatsächliche Möglichkeit, Angreifer permanent zu übertrumpfen!



Trusted Computing Group - CyRes

Grundprinzip: Fail-Safe und Wiederherstellung

- Erhöht die Verfügbarkeit des Systems
- Erhöht die Robustheit und Resilienz
- NIST definiert hierzu ein ganzes Framework, dass mit Recovery endet
- Eine vertrauenswürdige Wiederherstellungsoption besiegt praktisch jeden Angreifer auf Geräten
- Bedrohungsanalyse, Schutzmaßnahmen, Erkennung und Abwehr sind jedoch integral
  - Müssen nach Angriff überarbeitet werden!



NIST „The Five Functions“ (2021)

„A system must be able to operate and fulfill its primary mission under adverse conditions.“—NIST

Ein paar praktische Probleme…

- Wiedersprüche sind nicht zu verstehen als „gut gegen schlecht“:
  - Gilt nur in der „echten“ Welt in der Entwicklung und im Betrieb von Systemen - die Forschung vertritt andere Ansichten.

Sieger	***Verlierer***
Bequemlichkeit	Sicherheit
Ressourcenteilung	Isolation
Bug-Fixes	Verifikation
Regelwerke	Garantien
Zugriffskontrollen	Informationsflusskontrolle

B. Lampson - Perspectives on Computer Security, 2015

Zusammenfassung

Was wir können

- Einfaches sehr gut absichern
- Komplexe Systeme durch Isolation schützen
- Ein „Sicherheitstheater“ inszenieren

Was wir (noch) nicht können

- Komplexe Systeme sicher gestalten
- Große Systeme absichern, wenn Isolation nicht möglich ist
- Sicherheit bei zu behalten, wenn sich System oder Umstände ändern
- Menschen zu guten Entscheidungen für IT-Sicherheit bringen
- Unsere Bedingungen an „Privatheit“ verstehen

B. Lampson - Perspectives on Computer Security, 2015

Referenzen

- CIA - Security Controls for Computer Systems, Ware / RAND Corp, 1970.
- B. Lampson - Perspectives on Computer Security, 2015
- B. Lampson - Formal Methods for Design, 2006
- Raluca Ada Popa - Principles of Computer Security, 2020