FHV

IT-Sicherheit

Secure Software Engineering

Teil 1

Armin Simma

SS 2016

Literatur

- Paulus, Sachar (2011): Basiswissen Sichere Software: Aus- und Weiterbildung zum ISSECO Certified Professional for Secure Software Engineering. Auflage: 1. Heidelberg, Neckar: Dpunkt Verlag.

- Dieser Foliensatz basiert grossteils auf obigem Buch.

Übersicht Foliensatz

- Einleitung: die Sicht des **Kunden**
- **Methodologien** für sichere SW (Entwicklungsmethoden)
- Sicherheits**anforderungen**
- **Bedrohungsmodellierung**

Übersicht Foliensatz

- **Einleitung: die Sicht des Kunden ←**
- Methodologien für sichere SW (Entwicklungsmethoden)
- Sicherheitsanforderungen
- Bedrohungsmodellierung

Interessengruppen

- oft gibt es verschiedene Interessengruppen/ Beteiligte
  - öffentliche Hand, Banken, Versicherungen, Benutzer, Führungskräfte
- **multilaterale/ mehrseitige Sicherheit**: Versuch **alle** Sicherheitsanforderungen der Gruppen zu erfüllen
- Achtung: zum Teil gibt es Konflikte in den Anforderungen; **vorher klären**

(Sicherheits-)Anforderungen des Kunden

- Für die “einfache” Praxis: Der Kunde kennt/weiß seine (Sicherheits-)Anforderungen oft selber nicht

→ **aktiv** die Anforderungen aus dem Kunden herausarbeiten

  - (Sicherheits-)Anforderungen nur implizit beim Kunden (“im Kopf”)
  - (Sicherheits-)Anforderungen sind sehr individuell auf Anwendungsszenario bezogen
    - Bsp. Zeitaufzeichnungs-Sytem
  - **Anwendungsszenario** mit Kunden durchgehen, dort auf Schutzbedarf eingehen

Typische Anforderungen

- Vertraulichkeit von personenbezogenen Daten
- Vertraulichkeit von sensiblen (sensitive) Firmendaten
- Verfügbarkeit und Integrität der Informationen
- Verfügbarkeit und Integrität des Systems auf dem die SW läuft (inkl. SW selbst)
- Zugriffsschutz (Berechtigungssystem): Abwehr von Insider-Attacken
- Robustheit gegen Angriffe von außen

Klassifikation von Sicherheitserwartungen an Produkte

- Dem Kunden ist normalerweise egal, wie seine Anforderungen erfüllt werden, durch z.b.
  - sichere SW selbst
  - oder Zusatzlösung z.b. Firewall
- Kosten-Nutzen-Analyse: **vorher**! bei Sicherheit: Risiko-Nutzen-Kosten.
- Kunde hat meist gewisse Risikobereitschaft

Ansätze: Sicherheitsanforderungen erfüllen durch:

1. sichere Funktionen: SW weist bestimmte Sicherheitseigenschaften auf.
   - z.b. NW-Protokoll-Schnittstellen so geschrieben, dass Angriff keine Wirkung zeigt
   - keine BufferOverflows etc. möglich
2. Sicherheitsfunktionen
   - eingebaute Firewall; eingebaute Sicherheitschecks (z.b. Validierung)
3. sichere Konfiguration
   - Konf. möglich; Abschalten aller nicht erforderlicher Protokolle
4. Zusatz-Lösungen
   - externe (z.b. Firmen-)Firewall

Klassifikation von Sicherheitserwartungen an Produkte

- Oft auch Kombination der 4 Klassen
- Frühzeitig mit Kunden abstimmen, welche Lösung/Kombination für ihn möglich ist
- Nicht selber (als Hersteller) festlegen , welche Lösung a priori verwendet werden muss.

Klassifikation von Sicherheitserwartungen an Hersteller

- betrifft SW-Entwicklungsfirma selbst wie auch Zulieferer

- Wie geht Hersteller mit Schwachstellen um? Prozesse dafür? Info an den Kunden? Standards für Schwachstellenbeschreibungen eingehalten?
- Wie lange benötigt Hersteller , um Schwachstelle zu entfernen: Durchschnitt, Varianz, Max?
- Zertifiziert? Entwicklungsprozesse? Welche Produkte zert. ?
- andere Nachweise für Kompetenz in SecSW-Engineering

Compliance-Anforderungen

- Aktuell noch keine Anforderungen für Produktsicherheit direkt an SW (in Europa)
  - indirekt jedoch sehr wohl; durch Produkthaftungsgesetze
  - Hersteller muss im Schadensfall nachweisen, dass er das Produkt nach dem aktuellen “Stand der Technik” erstellt hat.
  - dazu gibt es besonders im Bereich **funktionale Sicherheit** (Safety!) einige Normen
  - IEC 61508 (E/E/PE); 26262 (Auto); ISO 14971 (Medizintech.)

- Für operationale Sicherheit im Unternehmensumfeld ergeben sich Sicherheits-Vorschriften aus KonTraG (Kontroll- und Transparenzgesetz); Handelsgesetzb; Aktiengesetz; TelekomG
- **Risikomanagement** ist verpflichtend -> daraus ergeben sich Vorschriften für Informationssicherheit
- im deutschsprachigen Bereich: Informationssicherheit meist mit IT-Grundschutz/ ISO 2700x

- Ganz aktuell (**Sommer 2015**): **IT-Sicherheitsgesetz**: Gesetz zur Erhöhung der Sicherheit informationstechnischer Systeme
  - Betreiber “kritischer Infrastrukturen” (Telekom, Kraftwerke…): besondere Sorgfalt; Mindeststandards
  - Bei Vorfall (*incident*, z.b. Angriff..) Meldepflicht

Compliance-Anforderungen

Gesetze/ Regeln im Umfeld:

- PCI-DSS: Payment Card Industry Data Security Standard
- SarbanesOxley (USA); EuroSOX
- HIPAA: Health Insurance Portability and Accountability Act: USA
- Nationale Datenschutzgesetze: grosse Unterschiede EU-USA
  - Safe Harbor; 2.Feb 2016: neues Datenschutz-Abkommen EU-USA

- aber auch für KMU und Freiberufler gibt es Gesetzl. Vorschriften:
- http://www.erfolg-als-freiberufler.de/it-sicherheit-datenschutz.php
  - Datenschutz der Kunden-/Mandanten-/Patienten-daten
  - Buchhaltung (Manipulationssicher, Backups)

Compliance-Anforderungen

(Rechtliche) Konsequenzen eines Angriffs

Beispiel: Target:

- Klagen gegen Target (Dez. 2013: bereits 40 Klagen)
  - (1) Target failed to act reasonably in adopting safeguards that would have prevented the breach from happening; and/or,
  - (2) Target didn't notify affected consumers quickly enough.
- Austausch von Kreditkarten
- Regierungs-Untersuchungsausschüsse
- http://ivebeenmugged.typepad.com/my_weblog/2014/02/target-consequences.html
- http://www.datasecuritylawjournal.com/2013/12/30/the-target-data-breach-lawsuits-why-every-company-should-care/
- http://cblr.columbia.edu/archives/13179

Compliance-Anforderungen

(Rechtliche) Konsequenzen eines Angriffs

Beispiel: Sony:

- http://blog.eiqnetworks.com/blog/bid/359151/Financial-Consequences-of-Sony-s-Data-Breach
- z.b. Klagen der “Screen Actors Guild”

- https://www.riskbasedsecurity.com/2014/12/a-breakdown-and-analysis-of-the-december-2014-sony-hack/

Perspektiven von SW-Sicherheit

1. **Angriffsorientiert**: Angriffe mögl? Welche? -> “Binäres” Resultat (sicher/ unsicher)
   - Nachträgliches Hinein**testen** von Sicherheit

2. **Technologie-orientiert**: perfekte Absicherung; z.B: formaler Beweis; Trusted Computing
   - funktionale Sicherheit
   - hochsichere Systeme (Militär, kritische Infrastrukturen z.b. Kraftwerke)
   - Es fehlt **Angemessenheit:** für viele Systeme ist formaler Beweis etc. zu teuer

3. **Werte-orientiert**

Perspektiven von SW-Sicherheit

3. **Werte-orientiert**
   - Für die meisten Systeme der ideale Ansatz
   - Wirtschaftlichkeit! *Economic Security*
     - Lohnt sich die Maßnahme? Ist das Ziel verlockend genug? Wahrscheinlichkeit?
     - Nur die Gegenmaßnahmen finanzieren, die sich auch rentieren!
     - Risiko-orientiert

   - Welches sind **Werte des Kunden**?
   - Wie die Werte ermitteln?
     - Use Cases -> ergeben meist auch Werte
       Weiters: Misuse Cases -> mögliche Bedrohungen;
       Risiko abschätzen
     - Klassifizieren der Werte
       Informationen, Güter, Know-How, Funktionalität
       Klassifikation -> Checkliste

Vgl: ISMS(1) (z.b. 27001)
Werte – Bedrohungen – Risko
https://www.sicherheitshandbuch.gv.at/siha.php
(1) Informationssicherheitsmanagementsystem

Von Erwartungen zu Anforderungen

- Requirements Engineering
- Anforderungsmanagement

- Wichtig:
  - Immer mit Kunden gemeinsam
  - nicht auf technischer Eben, sondern auf Anwendung(szenarien) bezogen
    - Use-Cases helfen oft

Von Erwartungen zu Anforderungen bei Sicherheitsanfordrungen

- Großer Unterschied zu “normalem” Requirements Engineering:
  - **Kunde** sieht (oft) keinen Mehrwert in Sicherheitsanforderungen
  - Kunde will sich damit (meist) nicht auseinandersetzen
  - Für Kunde ist Sicherheit ein technisches Umsetzungs-Problem des Herstellers

- Wie als **Hersteller** damit umgehen?
  - Viele Hersteller (besonders im Projektgeschäft) machen “Minimalvariante” für S.
  - Problem dabei: wenn (kritische) Schwachstelle später bekannt wird, fällt das auf den Hersteller zurück
  - -> Image!

- Da man also als Hersteller sowieso Sicherheit ernst nehmen sollte:
- Gleich Kunde mit einbeziehen!

Wenn der Hersteller dem Kunden hilft, seine Sicherheitsanforderungen zu definieren, hilft er sich selbst.

Übersicht Foliensatz

- Einleitung: die Sicht des Kunden
- **Methodologien für sichere SW (Entwicklungsmethoden)** ←
- Sicherheitsanforderungen
- Bedrohungsmodellierung

Softwareentwicklungsmethoden

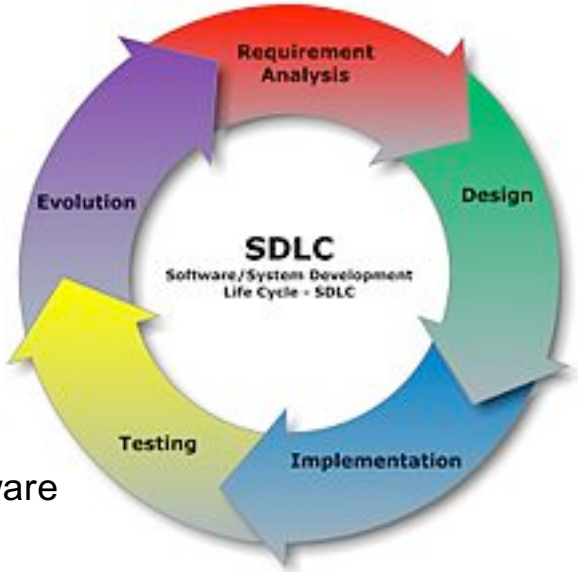
Methodologien für sichere SW

- 2 Aspekte

1. Sichere Entwicklungsumgebung
   - Gefährdung während Entwicklung begegenen:
   - gesicherte physische Umgebung, sicheres OS, Entwicklungsumgebung (Nachvollziehbarkeit von Entscheidungen!)
2. Methodologie für sichere SW-Entwicklung
   - Gesamtprozess
   - Sicherheits-Aktivitäten in jeder Phase

SW-Prozess

- Neu bei Sec.Engineering:
  Risiko- und Bedrohungsanalyse; Schutzziele, Sicherheitsanforderungen definieren



Aus: Wikibooks: “Introduction to Software Engineering/Process/Life Cycle”

http://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Process/Life_Cycle

Softwareentwicklungsmethoden: (Prozess)-Modelle

- Wasserfall- , V- Spiral-Modell
- Prototypgetriebene
- stark strukturiere Methoden: z.b. Rational Unified Process
- agile Methoden: Scrum, EP, Test Driven Development

- grundsätzlich alle Modelle möglich (Prototyp eher weniger)
  - bei einigen Modellen müssen zusätzliche Aktivitäten gesetzt werden

- **Beim bekannten bzw. bisher verwendeten Modell bleiben; um Sicherheitsberührungspunkte anreichern!**

Softwareentwicklungsmethoden: (Prozess)-Modelle

- Wasserfall- , V-Modell
  - gut geeignet; wichtig jedoch dass Iterationen (feedback-Schleifen) vorhanden sind
  - Sicherheit passt gut ins V-Modell , da enge Kontrolle der Anforderungen durch Tests
- Spiralmodell
  - neue Schleife entsteht meist durch Verbesserungsanforderungen (mangelnde Qualität)
  - passt gut mit Sicherheit zusammen
- Prototypgetriebene
  - schlecht geignet: man weiss nicht, wann welches Umfeld nun gültig ist; somit auch nicht welche Bedrohungen/ Sicherheitseigenschaften
- stark strukturiere Methoden: z.b. Model Driven Development
  - Modell ist relativ starr
  - wichtig: in frühen Phasen bereits klare Sicherheitsanforderungen
- Rational Unified Process
  - ähnlich MDD
  - da RUP aber auch Iterationen beinhaltet: vereinfacht Sicherheitsaspekte gut zu berücksichtigen.

Softwareentwicklungsmethoden: (Prozess)-Modelle

- agile Methoden:
  - Hohe Flexibilität, kurze Zyklen, die jeweils formal nach klaren Regeln ablaufen
  - Qualität der entwickelten Funktionalitäten kann gut nachvollzogen/sichergestellt werden
    - Daher auch Sicherheitsfunktionen gut bis sehr gut realisiert
  - Problematisch: **nicht-funktionale** Anforderungen
  - -> Sicherheitseigenschaften außerhalb des Sprint-Rhythmus verwalten und ueberpruefen!
    - mit speziellen Maßnahmen

Maßnahmen zur Verbesserung Ergänzung des Entwicklungsprozesses/ **Prinzipien:**

- Angewandtes Risikomanagement
  - Kosten-Nutzen-Risiko
  - Folgekostenschätzung oft schwierig (Imageverlust etc.)
  - Beginn bei den (Risiko-)größten Schwachstellen
- Wissen:
  - System muss (gut) bekannt sein; Architektur, Umfeld etc.
  - daher immer (Domänen-)Experten, Architekten etc. mit einbinden
- Etablierte Lösungen verwenden
  - z.b. Security Patterns (Sicherheitsmuster)

Maßnahmen zur Verbesserung Ergänzung des Entwicklungsprozesses/ **Prinzipien:**

- Proaktivität
  - oft wird versucht ganz am Ende (mit z.b. Penetrationstests etc.) Sicherheit in SW hineinzutesten
  - Je früher, desto billiger
  - besonders, wenn **Architektur** bereits Sicherheitsmängel hat

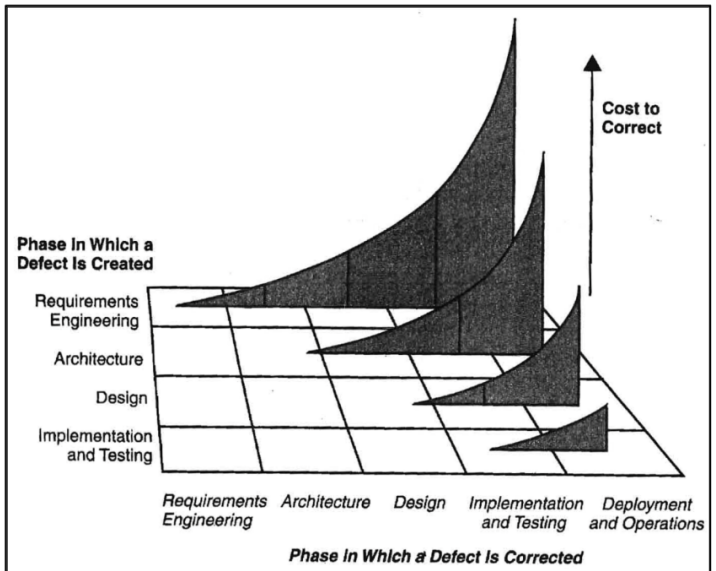


Figure 1–4: *Cost of correcting defects by life-cycle phase*

Quelle: Software Security Engineering, Allen et al

Prinzipien/Phasen für sichere SW-Entwicklung

1. Anforderungsanalyse, -definition;
   - ◇ Sicherheitsanforderungen
2. Sicherer SW-Entwurf
   - ◆ Bedrohungsanalyse;
   - ◆ Prinzipien für sicheren SW-Entwurf (z.b. Saltzer und Schroeder)
   - ◆ Sicherheits-Muster
3. Sicheres Codieren
   - ◇ Techniken für sicheres Programmieren (z.b. OWASP, “24 deadly sins”)
4. Tests (inkl. Sicherheitstests)
   - ◆ Penetrationstests
   - ◆ sicherheitsrelevante Testfälle
5. SW-Einrichtung:
   - ◇ sichere Konfiguration
   - ◇ sicherer Betrieb
6. Security-Response
   - ◆ Umgang mit Schwachstellen

Alle diese Phasen müssen begleitet werden, damit Vorgaben auch eingehalten werden!

Warum sind alle Phasen wichtig?

- Warum sind alle Phasen wichtig?
- Viele Programmierer glauben: “Wenn beim Codieren alles richtig gemacht, ist SW sicher”. “Gib mir Tipps, wie sicher programmiert wird und SW wird sicher”

- Eigene Erfahrung (Studierende/r fragt mich: welches Authentifzierungsverfahren? “TLS/SSL hab ich ja auch verwendet”; Welcher Algo (AES 256, Hash-Algo SHA3…?
- Nach längerer Diskussion komm ich drauf: Es wird Clientseitiges Java-Script verwendet. Angreifer hat somit volle Kontrolle auf Code.

- Weiteres Beispiel: Hilfe-”Männchen” von Microsoft: “Popt” immer auf, wenn Benutzer was falsches eingibt, nicht weiter weiss etc.
- Bei Login-Screen: Männchen empfiehlt: Wenn Passwort vergessen: neues Passwort eingeben.

Übersicht Foliensatz

- Einleitung: die Sicht des Kunden
- Methodologien für sichere SW (Entwicklungsmethoden)
- **Sicherheitsanforderungen** ←
- Bedrohungsmodellierung

Anforderungsanalyse

- Anforderungsanalyse:
  - gute Anforderungen sehr wichtig auch für Sicherheit
- Unterscheidung funktionale – nicht funktionale Anforderungen
- **funktionale Anforderungen**
  - beziehen sich auf eine bestimmte Funktionalität: meist Input-Verarbeitung –> Output
  - **WAS** soll das Produkt/ SW tun?
- **Nicht-funktionale Anforderungen**
  - beziehen sich auf eine Eigenschaft:
  - Wie?
  - z.b. Produkt ist sicher, schnell, benutzerfreundlich

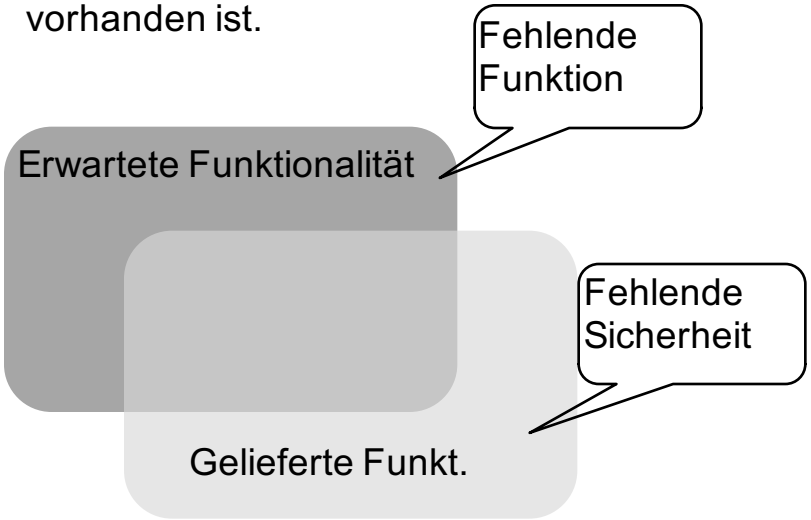
Anforderungsanalyse: Wie kommt man zu (guten) Anforderungen?
allgemeines (nicht sec-bezogen) Requirements Engineering

- Glossar
- Anwendungsbereich definieren: Umgebung
- Interessengruppen identifizieren
  - Jede Anforderung sollte Eigner haben
  - Bsp: Kunden, Endbenutzer, Entwickler, Markt
- Anforderungen definieren
  - Quellen: ähnliche Produkte; Szenarien; Audits
- Dokumentieren!
- Akzeptanzkriterien definieren
- Prüfung der Anforderungen
  - klar, eindeutig , widerspruchsfrei

vgl.: Sachar Paulus: Basiswissen sichere SW

Anforderungsanalyse: Sicherheit

- Sicherheitsanforderungen sind oft nicht-funktionale
- Sicherheitsanforderungen bedeuten oft, dass eine bestimmte (nicht gewollte) Funktion **nicht** vorhanden ist.



vgl.: Sachar Paulus: Basiswissen sichere SW

"**difficult** to verify (i.e., instead of ensuring that *n* functions work properly, security has to ensure that virtually infinite unwanted functions=attack do not work) "

Anforderungsanalyse: Sicherheit

- funktionale Anforderungen relativ leicht zu testen
  - (nämlich einfach Funktion ausführen; dann Ergebnis mit Erwartung vergleichen)
  - daher **Akzeptanzkriterien** einfach zu definieren

- **Akzeptanzkriterien** für nicht-funktionale Anforderungen schwierig

- Hinweis: Akzeptanzkriterium != Testfall
  - A. beschreibt nur **Bedingung** unter denen Anforderung erfüllt ist

vgl.: Sachar Paulus: Basiswissen sichere SW

Anforderungsanalyse: Sicherheit

- **Akzeptanzkriterien** für **nicht-funktionale** Anforderungen
  1. Abbildung der nicht-funktionalen Eigenschaft auf eine funktionale Anforderungen
     - statt “Alle Anfragen von außen – ausser ssh - werden geblockt”
     - -> “Es gibt eine Netwerkkomponente (SW), die alle Anfragen von aussen überprüft. ssh-Anfragen werden durchgelassen, alle anderen verworfen”.
  2. Abbildung der nicht-funktionalen Anforderung auf eine (leichter überprüfbare) funktionale Anforderung plus eine nicht-funkt. A.
     - statt “keine Eingabe erlaubt Schadcode”
     - -> “es gibt Funktion, die Eingaben prüft…” (+ genaue Beschreibung der Prüfung, was ist erlaubt etc.)

       + “jede Funktion ruft vorher Prüffunktion auf; ist wiederum nicht-funkt.)
  3. Man erlaubt gewisse Ungenauigkeit; man geht nach spezifizierten Anzahl von Prüfungen davon aus, dass Eigenschaft erfüllt ist.
     - statt “Die Website ist von Hackern nicht angreifbar”
     - -> die 1000 aktuellsten Angriffe werden abgeblockt

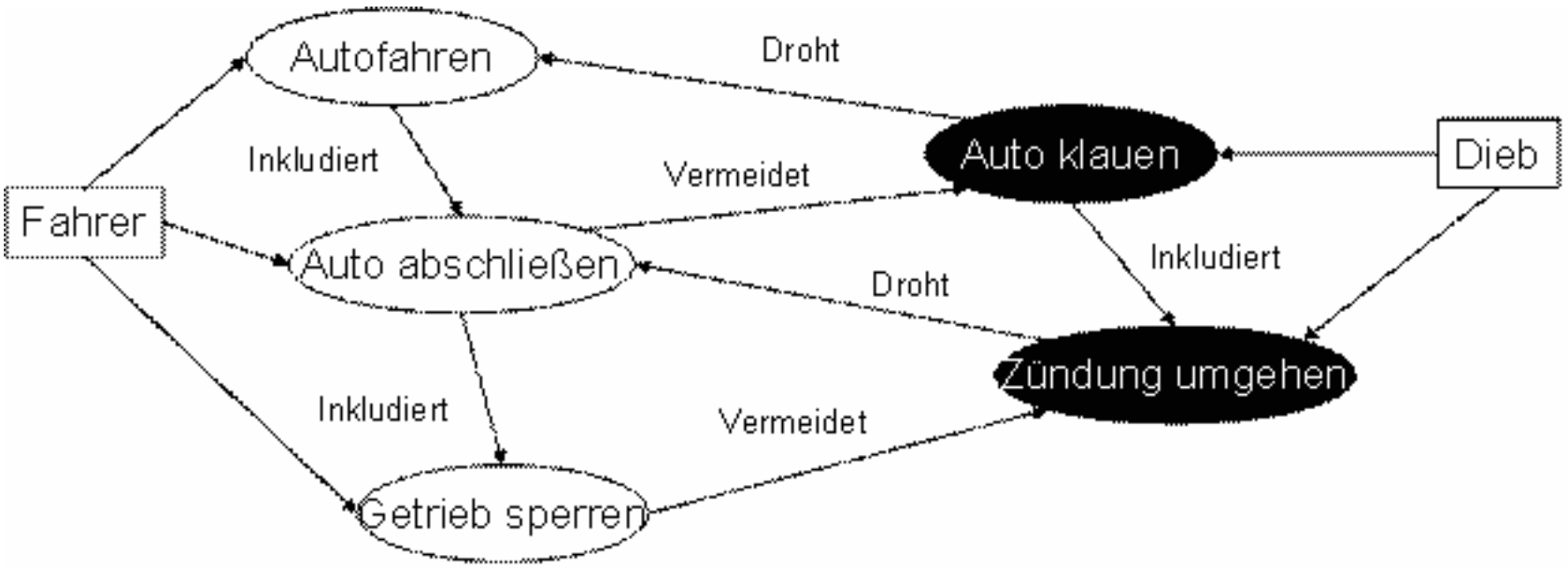
vgl.: Sachar Paulus: Basiswissen sichere SW

Wie kommt man zu Sicherheits-Anforderungen?

- Grundwerte: CIA
- Misuse Cases
- Kontext/ Anwendungsbereich betrachten
  - z.b. nur intern, lokal; im/ über Internet; Cloud
- Bedrohungsmodellierung
- Auflösung von Konflikten
  - oft Konflikt mit Usability, Interoperabilität
  - Konflikte ansprechen!

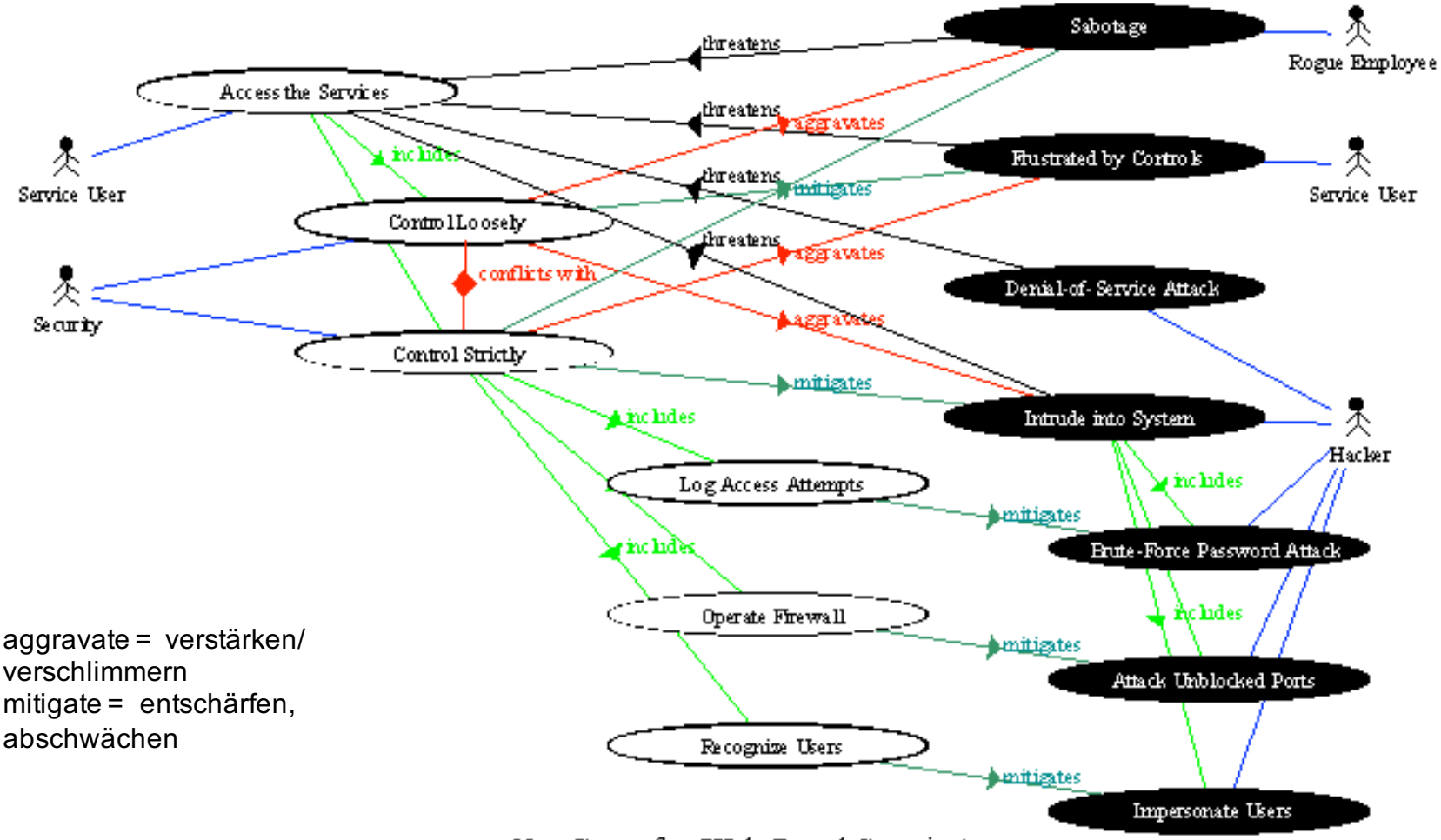
vgl.: Sachar Paulus: Basiswissen sichere SW

Misuse Cases



Quelle: http://www.re-wissen.de/opencms/Wissen/Techniken/Erhebung_von_MisUse_Cases.html

Misuse Cases



Use Cases for 'Web Portal Security'

aggravate = verstärken/ verschlimmern
mitigate = entschärfen, abschwächen

Quelle: http://www.scenarioplus.org.uk/papers/misuse_cases_hostile_intent/misuse_cases_hostile_intent.htm

Wie kommt man zu Sicherheits-Anforderungen?

- Standards

CC Common Criteria

- Security Audit (FAU): *Logging*
- Communication (FCO): *Identification of parties, repudiation*
- Cryptographic Support (FCS): *Cryptography*
- User Data Protection (FDP): *Access control*
- Identification and Authentication (FIA)
- Security Management (FMT): *System security & mgmt.*
- Privacy (FPR): *Anonymity, pseudonymity*
- Protection of the system security functions (FPT)
- Resource Utilization (FRU): *Utilization limitations*
- System Access (FTA): *Access/connection limits*
- Trusted path/channels (FTP): *Secure channels, sockets*

vgl.: Sachar Paulus: Basiswissen sichere SW

Wie kommt man zu Sicherheits-Anforderungen?

- Standards

OWASP Themenfelder

- 8 Handling E-Commerce Payments
- 9 Phishing
- 10 Web Services
- 11 Ajax and Other "Rich" Interface Technologies
- 12 Guide to Authentication
- 13 Guide to Authorization
- 14 Session Management
- 15 Data Validation
- 16 Interpreter Injection
- 17 Canonicalization, locale and Unicode
- 18 Error Handling, Auditing and Logging
- 19 File System
- 20 Distributed Computing
- 21 Buffer Overflows
- 22 Administrative Interface
- 23 Guide to Cryptography

vgl.: Sachar Paulus: Basiswissen sichere SW

Wie kommt man zu Sicherheits-Anforderungen?

- weitere inhaltliche Bereiche:

Sicherheitskernanforderungen:

- Rollenkonzept
- Zugriffskontrolle
- Identitätsmgmt
- AAA
- Interoperabilität
- Informationsschutz
- PII
- Audit-Funktionen
- Schutz der verschiedenen Angriff(stypen)
- Verfügbarkeit
- Integrität der Prozess..

vgl.: Sachar Paulus: Basiswissen sichere SW

Wie kommt man zu Sicherheits-Anforderungen?

- weitere inhaltliche Bereiche:

Sicherheitsaspekte der Usability:

- wie viel/ welche Infos werden dem Endbenutzer angezeigt
  - aus Sicherheitsgründen besser: wenig
- “gute” Usability
  - viele Problem entstehen durch Unverständnis -> falsche Entscheidung des Users
  - Beispiel Zertifikatswarnung des Browsers
  - Lösung:
    - oft hilft es Kontext zu betrachten
    - nur die Infos anzeigen , für die er überhaupt Berechtigung hat

Einsatzumgebung

- Bedingungen (HW, SW) am Einsatzort? (z.b. Authentifizierung?) Netzwerkumfeld?

vgl.: Sachar Paulus: Basiswissen sichere SW

Übersicht Foliensatz

- Einleitung: die Sicht des Kunden
- Methodologien für sichere SW (Entwicklungsmethoden)
- Sicherheitsanforderungen
- **Bedrohungsmodellierung** ←

Bedrohungsmodellierung

Bedrohungsmodellierung: Nutzen

- Werte werden identifiziert
  - auch (vor allem) aus Sicht des Kunden
- Verständnis der Angriffsmechanismen
  - sich in Rolle des Angreifers begeben
  - Strukturierung: Angriffsmuster verwenden (sonst zuviele mögl. Angriffe)
- risikobasierte Einschätzung
  - Investitionen werden priorisiert
- Ergebnisse der Modellierung: zusätzliche Anforderungen und Testfälle
- Designfehler werden identifiziert
  - ca. 50% der Schwachstellen im Design

Bedrohungsmodellierung: Ansätze

1. Angriffszentriert
2. Softwarezentriert
   - Daten, die von SW verarbeitet werde, im Mittelpunkt
   - besonders, wenn Verwendung (Umfeld) der SW noch nicht bekannt (z.b. bei Bibliotheken)
3. Wertezentriert
   - Fokus auf Werte

Bedrohungsmodellierung: Phasen

1. Werte (und Akteure) identifizieren
2. Überblick Architektur
3. System zerteilen
4. Bedrohungen identifizieren
5. Bedrohungen dokumentieren
6. Bedrohungen bewerten
7. Gegenmaßnahmen planen

- aus Paulus, Sachar (2011): “Basiswissen Sichere Software”
- Sehr ähnlich auch zu MS Threat Modeling:
  - J.D. Meier et al. (2003) “Improving Web Application Security: Threats and Countermeasures”
  - http://msdn.microsoft.com/en-us/library/ff648644.aspx

Bedrohungsmodellierung: Phasen

1. **Werte (und Benutzer) identifizieren**

- nicht nur Werte und Benutzer, sondern auch Aktionen und Operationen auf/mit Werten durch Akteure
  - Beziehung zwischen diesen ohne konkrete SW
- Fragen:
  - Was ist für Kunden (und dessen Kunden) am wichtigsten? Was ist für Benutzer am wichtigsten?
  - Was ist für Hersteller am wichtigsten?
  - Wer kann Werte beschädigen? Ziel? Wer hat Interesse an den Werten?
  - Schutzmassnahmen, die Gesetz vorgibt?

- Werte müssen nicht unbedingt Entitäten des SW-Modells (Architektur) sein
  - Bsp: Image
- Vergleiche ISMS (z.b. ISO 2700x) : Werte quantifizieren; hier jedoch schwierig

Bedrohungsmodellierung: Phasen

2. Überblick Architektur

- Henne-Ei-Problem: Architektur/ Design existiert noch gar nicht
  - Bedrohungsmodellierung benötigt (grobe) Architektur
  - Architektur richtet sich (auch) nach den Sicherheitsvorgaben

➔ ideal: iterativer Prozess

- Wichtig für Bedrohungsmodellierung
  - Datenhaltungen und
  - Datenflüsse

➔ daraus können **Vertrauensgrenzen** (trust boundaries) abgeleitet werden

Bedrohungsmodellierung: Phasen

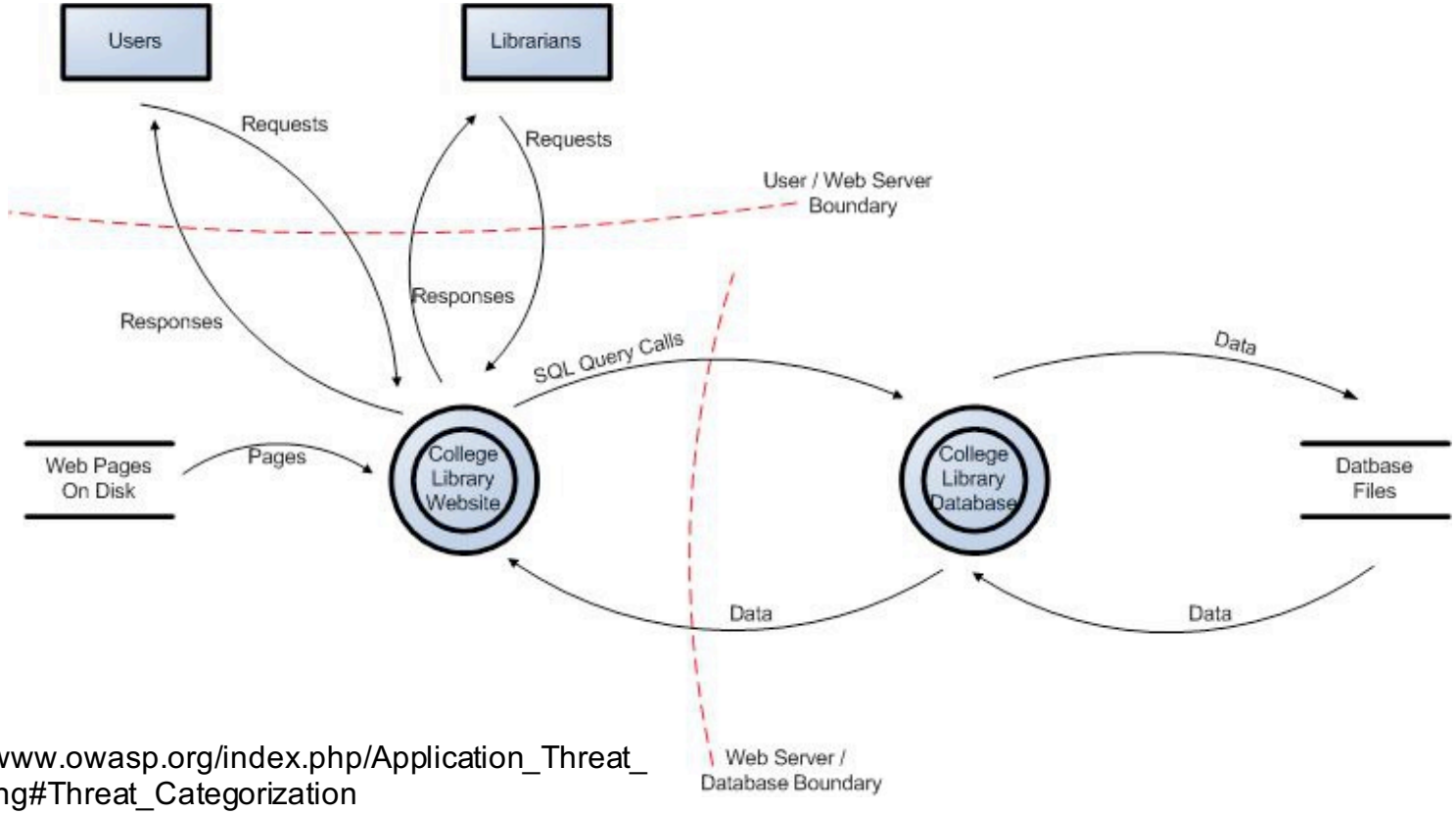
2. **Überblick Architektur**

➔ **Vertrauensgrenzen** (trust boundaries)

- Vertrauensgrenze trennt Entitäten/Kontexte, die sich nicht “blind” vertrauen und damit Daten nur unter Voraussetzungen teilen
- Innerhalb eines Kontextes (ohne Grenze) werden alle Daten geteilt
  - Analogie/ vergleiche: Shared Memory versus z.b. virtueller Speicher bzw. Message Queues

- Je mehr Vertrauensgrenzen desto sicherer (im Prinzip), aber desto aufwändiger
- Optimale Anzahl von Vertrauensgrenzen finden: ausreichend hohe Sicherheit; aber handhabbar

Bedrohungsmodellierung: Phasen: **Vertrauensgrenzen**



- Quelle: https://www.owasp.org/index.php/Application_Threat_Modeling#Threat_Categorization

Bedrohungsmodellierung: Phasen

2. **Überblick Architektur**

- An den Vertrauensgrenzen werden Angriffe/ Angriffsklassen ausprobiert
  - Z.b. nach CIA oder STRIDE

Bedrohungsmodellierung: Phasen

2. **Überblick Architektur**

- **Datenhaltungen:**
  - Datenbank, Webserver, Browser, Dateien (dateisystem)
  - in feingranularem Modell: Klassen

- **Datenflüsse**
  - Kommunikation/Transfer von Daten
  - Übergabe zwischen Methoden (durch Parameter)

- Architekturentwurf: Datenhaltungen, Datenflüsse, Werte, Akteure (aus Phase 1)

- Ziel: Überblicksbild der Zusammenhänge zwischen den 4 Entitäten
  - Schnittstellen! Weil Angriffe meist an Schnittstelle passieren

Bedrohungsmodellierung: Phasen

3. **System zerteilen**

- an den Vertrauensgrenzen teilen und
- einzelne Kontexte (innerhalb Vertrauensgrenze) feiner modellieren
- Wozu?
  - richtige Granularität. (Oft schwierig)
  - Angriffspunkte innerhalb Vertrauensgrenze finden

- nicht zu viele Iterationen

Bedrohungsmodellierung: Phasen

1. Werte (und Akteure) identifizieren
2. Überblick Architektur
3. System zerteilen
4. **Bedrohungen identifizieren** ←
5. Bedrohungen dokumentieren
6. Bedrohungen bewerten
7. Gegenmaßnahmen planen

Bedrohungsmodellierung: Phasen

4. **Bedrohungen identifizieren**

- Wichtigste Phase
- erster Schritt: möglichst viele Bedrohungen ermitteln

- Hilfsmittel:
  - Abuse und Misuse Cases,
  - Brainstorming
  - DFDAnalysen,
  - Fragenkataloge,
  - Threat Profiling
  - …

Bedrohungsmodellierung: Phasen

- **Bedrohungen identifizieren**

- Quelle: Matthias Rohr : Sicherheit von Webanwendungen in der Praxis

Kürzel	Beschreibung	Anspruch
Integration existierender Findings	Integration von Findings früherer Sicherheitsuntersuchungen, z. B. Pentests, Codeanalysen oder anderen Bedrohungsmodellen/-Verzeichnissen	Gering
Identifikation von Known-Issues	Identifizierung bekannter Sicherheitsprobleme im eingesetzten Technologiestack (z. B. mittels Internetrecherche)	Gering-Mittel
Brainstorming	Diskussion möglicher Bedrohungen mit unterschiedlichen Stakeholdern bzw. Fachexperten (SMEs), z. B. im Rahmen eines Workshops	Gering-Mittel
Fragenkataloge	Beantwortung von Fragen, die sich auch aus bestimmten Anwendungsfällen oder verwendeten Technologien ableiten lassen. Beispiele: – Existieren bereits bekannte Bedrohungen für System/Anwendungsfall/Technologie? – Existieren bestimmte Aktionen, die von bestimmten Anwendergruppen niemals ausgeführt werden sollen? – Wie hoch ist der Schaden, wenn einzelne Funktionen temporär ausfallen oder Daten kompromittiert werden? – Wie zuverlässig (bzw. fälschungssicher) müssen Benutzeridentitäten festgestellt werden? – Wie hoch muss das Vertrauen in den erstellten Code sein?	Gering
Threat Profiling	Mittels Threat Profiling lassen sich fachliche oder technische Attribute auf relevante Bedrohungen abbilden. Erfüllt eine Webanwendung etwa eine bestimmte fachliche Aufgabe (z. B. Admin-GUI, Shop-System, Web-Mailer oder Bewerber-Portal) so lassen sich daraus oft recht umfangreiche fachliche Bedrohungsprofile ableiten, die sich auf eine entsprechende Anwendung einfach anwenden lassen. Das gleiche gilt für technische Aspekte einer Anwendung. Dies funktioniert auch für ganze Technologiestacks (z. B. spezifische Bedrohungen für Webanwendungenallgemein, oder SharePoint-Anwendungen im Besonderen), die sich auf technische Bedrohungsprofile abbilden und auf bestimmte Anwendungstypen beziehen lassen	Gering

Bedrohungsmodellie
Phasen

- **Bedrohungen identifizieren**

- Quelle: Matthias Rohr : Sicherheit von Webanwendungen in der Praxis

Kürzel	Beschreibung	Anspruch
Misuse Cases (fachlicher Bedrohungsfall)	Beschreiben eine mögliche missbräuchliche Nutzung bestimmter Anwendungsfälle (Use Cases). Misuse Cases lassen sich sowohl in grafischer als auch textueller Form für sicherheitsrelevante Anwendungsfälle (z. B. Anmeldung) beschreiben. In Abschn. 4.6.2 wurde bereits veranschaulicht, wie sich diese durch UML-Diagramme darstellen lassen. Bereits bei der Spezifikation von Anwendungsfällen können entsprechend geschulte Mitarbeiter mögliche Misuse Cases identifizieren, die im Rahmen einer Bedrohungsanalyse genauer analysiert werden	Gering-Hoch
Abuse Cases (fachlicher Bedrohungsfall)	Beschreiben mögliche Missbrauchsszenarien aus der Sicht einer bestimmten Bedrohungsquelle (z. B. eines Hackers oder aber auch eines internen Angreifers). Anders als Misues Cases bezieht sich ein Abuse Case nicht auf einen konkreten Anwendungsfall. Abuse Cases lassen sich sehr gut im Rahmen des Anforderungsmanagements abfragen; Beispiel: „Welche Informationen darf Rolle X nicht sehen/ was darf diese nicht tun können?“	Gering-Hoch
Analyse von Angriffsbäumen (Attack Trees)	Analyse potentieller Bedrohungen anhand Bewertung einzelner Angriffsvektoren	Mittel
Analyse des Rollen- und Berechtigungs-modells	Analyse des Rollen- und Berechtigungsmodells auf mögliche Bedrohungen in Hinblick auf Seperation of Duties, Need to Know, Need to Do etc. (Abschn. 3.12.4)	Mittel-Hoch
DFD-Analyse	Datenflussbasierte Analyse etwa auf Basis des STRIDE-Models (Microsoft Ansatz) oder durch eine Trust-Boundary-Analyse	Mittel-Hoch
Expertenanalyse	Informelle Analyse durch Sicherheitsexperten	Hoch

Bedrohungsmodellierung: Phasen

4. Bedrohungen identifizieren

- Wichtigste Phase
- erster Schritt: möglichst viele Bedrohungen ermitteln

- Hilfsmittel:
  - Brainstorming
  - Checklisten
  - Datenflussdiagramme (DFD)
  - Angriffsbäume/ Bedrohungsbäume
  - SW-Tools

Bedrohungsmodellierung: Phasen

4. **Bedrohungen identifizieren: Brainstorming**

- Art “Penetrationstest an der ersten High-Level-Architektur”
- “Wie würden Sie vorgehen, wenn Sie ihr System angreifen wollten?”
  - System = SW, so wie vorläufig in High-Level-Architektur geplant
- Team sollte mind. SW-Entwickler (Architekt, Programmierer) beinhalten

- typisch bei Brainstorming

1. Sammeln ALLER Ideen (keine Kritik, Wertung etc.)
2. Sortieren und Bewerten

- Unterstützung/Vorbereitung (der Teilnehmer) fürs Brainstorming
  - HAZOP (hazard and operability study)
  - STRIDE

Bedrohungsmodellierung: STRIDE

- **Klassifizierung von Bedrohungen mit STRIDE**

- entwickelt von MS

- spoofing: Erschleichen/Vorgabe anderer/falscher Identitäten
- tampering: Manipulation
- repudiation: Abstreiten
- information disclosure: Veröffentlichen von (sensitiven, vertraulichen) Informationen
- denial of service: Angriff auf Verfügbarkeit
- elevation of privilege: Verschaffen von zusätzlichen Rechten, meist: Admin-Rechte

[http://msdn.microsoft.com/en-us/magazine/cc163519.aspx]
http://msdn.microsoft.com/en-us/library/ee823878%28v=cs.20%29.aspx

- Wichtig ist für Brainstorming Phase 1: Klassifizierung soll kein Selbstzweck sondern Hilfe sein

Bedrohungsmodellierung: Phasen

4. **Bedrohungen identifizieren: Checklisten**

- gute Zusatzunterstützung zu anderen Methoden **aber**:
- Checklisten verleiten, das Hirn auszuschalten
- Checklisten sind oft spezifisch (für Branche, Produkt, Umgebung etc.)
  - Passt die Checklist wirklich auf meine SW?

- viele Checklisten listen nicht Bedrohungen, sondern (Schutz-)Massnahmen
  - meist einfach aus den Massnahmen die Bedrohungen abzuleiten.

Bedrohungsmodellierung

4. Bedrohungen identifizieren: Checklisten

- OWASP: “ASF Threat & Countermeasures List”
- https://www.owasp.org/index.php/CRV2_AppThreatModeling#Countermeasure_Identification

ASF Threat & Countermeasures List	
Threat Type	**Countermeasure**
Authentication	1. Credentials and authentication tokens are protected with encryption in storage and transit 2. Protocols are resistant to brute force, dictionary, and replay attacks 3. Strong password policies are enforced 4. Trusted server authentication is used instead of SQL authentication 5. Passwords are stored with salted hashes 6. Password resets do not reveal password hints and valid usernames 7. Account lockouts do not result in a denial of service attack
Authorization	1. Strong ACLs are used for enforcing authorized access to resources 2. Role-based access controls are used to restrict access to specific operations 3. The system follows the principle of least privilege for user and service accounts 4. Privilege separation is correctly configured within the presentation, business and data access layers
Configuration Management	1. Least privileged processes are used and service accounts with no administration capability 2. Auditing and logging of all administration activities is enabled 3. Access to configuration files and administrator interfaces is restricted to administrators
Data Protection in Storage and Transit	1. Standard encryption algorithms and correct key sizes are being used 2. Hashed message authentication codes (HMACs) are used to protect data integrity 3. Secrets (e.g. keys, confidential data) are cryptographically protected both in transport and in storage 4. Built-in secure storage is used for protecting keys 5. No credentials and sensitive data are sent in clear text over the wire
Data Validation / Parameter Validation	1. Data type, format, length, and range checks are enforced 2. All data sent from the client is validated 3. No security decision is based upon parameters (e.g. URL parameters) that can be manipulated 4. Input filtering via white list validation is used 5. Output encoding is used
Error Handling and Exception Management	1. All exceptions are handled in a structured manner 2. Privileges are restored to the appropriate level in case of errors and exceptions 3. Error messages are scrubbed so that no sensitive information is revealed to the attacker
User and Session Management	1. No sensitive information is stored in clear text in the cookie 2. The contents of the authentication cookies is encrypted 3. Cookies are configured to expire 4. Sessions are resistant to replay attacks 5. Secure communication channels are used to protect authentication cookies 6. User is forced to re-authenticate when performing critical functions 7. Sessions are expired at logout
Auditing and Logging	1. Sensitive information (e.g. passwords, PII) is not logged 2. Access controls (e.g. ACLs) are enforced on log files to prevent un-authorized access 3. Integrity controls (e.g. signatures) are enforced on log files to provide non-repudiation 4. Log files provide for audit trail for sensitive operations and logging of key events 5. Auditing and logging is enabled across the tiers on multiple servers

Bedrohungsmodellierung: Phasen

4. **Bedrohungen identifizieren: Checklisten**

- BSI: Sicherheit von Webanwendungen Maßnahmenkatalog und Best Practices Kapitel **“Maßnahmen und Best Practices”**
- BSI
- https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/WebSec/WebSec_pdf.pdf?__blob=publicationFile

- BSI: “Leitfaden Informationssicherheit. IT-Grundschutz kompakt”. Anhang: Checkliste
  - Achtung: Leitfaden ist nicht für SW sondern mehr für Betrieb von IT-Systemen, aber Ideen zu SW-Sicherheit können auch hier gefunden werden.

Bedrohungsmodellierung: Phasen

4. Bedrohungen identifizieren: Checklisten

- US-CERT: “Software Assurance Checklist for Software Supply Chain Risk Management”
- https://**buildsecurityin.us-cert.gov**/sites/default/files/SwA**Checklist**-v1.xlsx

- Einzelne Abkürzungen sind Referenzen auf Standards/ Normen: BSIMM, CMMI-ACQ, OSAMM ,PRM, RMM

- z.b. AM 1.1: “Build and maintain a top N possible attacks list. The SSG helps the organization understand attack basics by maintaining a list of the most important attacks. This list combines input from multiple sources: observed attacks, hacker forums, industry trends, etc. The list does not need to be updated with great frequency, and the attacks can be sorted in a coarse fashion. For example, the SSG might brainstorm twice per year to create lists of attacks the organization should be prepared to counter “now,” “soon,” and “someday.”

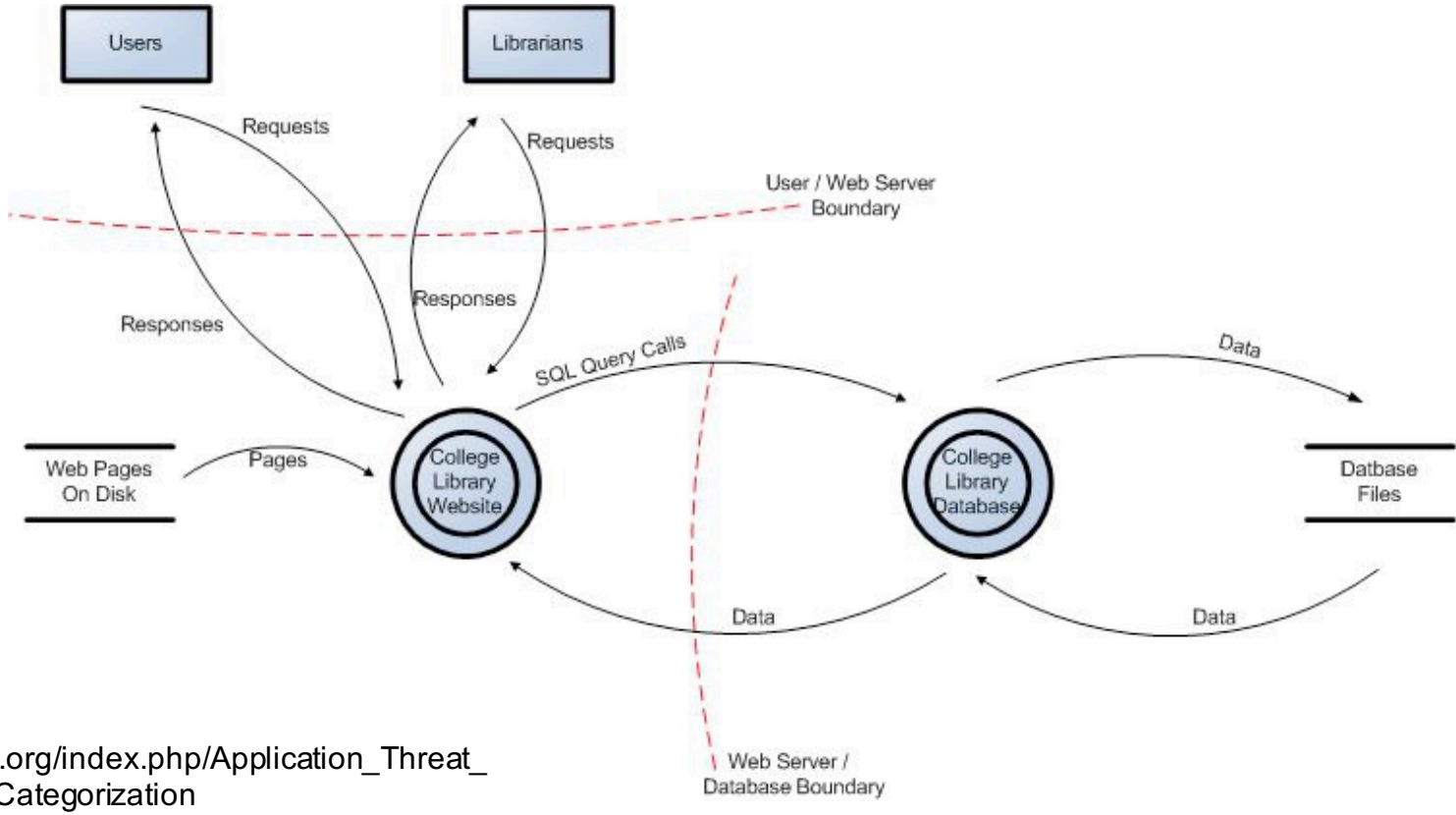
Threat Assessment
AM1.1
AM1.4
RSKM SG1 - SG2
-
TA1A
-
SG 3.2
-
RISK:SG1 - SG4
KIM:SG6
Identifies potential attacker profiles
AM1.3
-
RSKM SG1 - SG2
-
TA1B
-
SG 3.2
-
RISK:SG1 - SG4
KIM:SG6
Builds and maintains abuse cases and attack patterns
AM2.1
AM 2.2
RSKM SG2
-
TA2A

Bedrohungsmodellierung: Phasen

4. Bedrohungen identifizieren: Checklisten

- OWASP TOP 10
  - https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
  - https://www.owasp.org/index.php/Top_10_2013
- Howard, LeBlanc, Viega: "24 Deadly Sins of SW Security"

Bedrohungsmodellierung: Datenflussdiagramme (DFD)

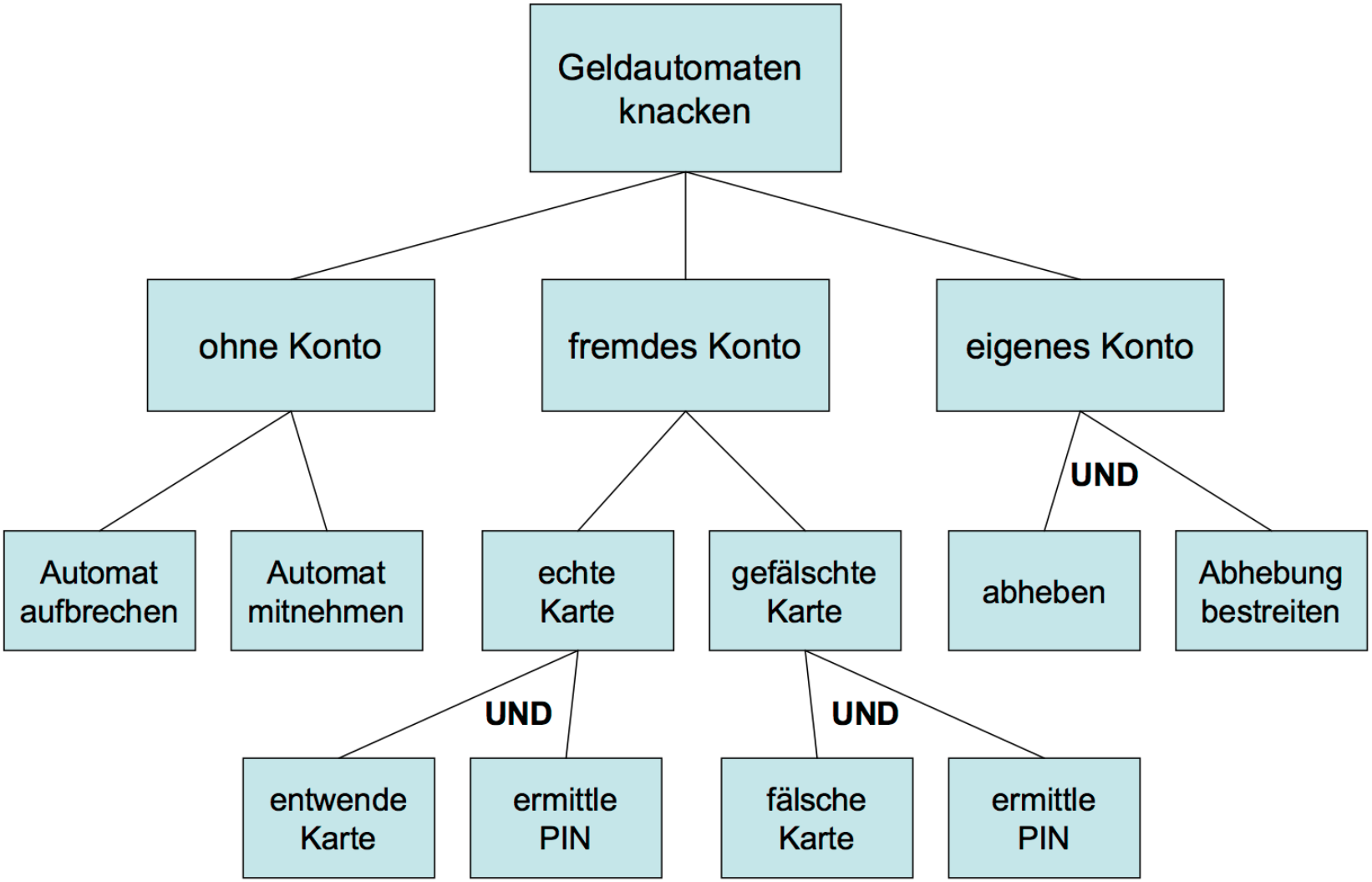


- Quelle: https://www.owasp.org/index.php/Application_Threat_Modeling#Threat_Categorization

Bedrohungsanalyse: **Bedrohungsbaum**

- Wurzel: konkrete Bedrohung bzw. Angriffsziel
- Knoten: Teilziel zur Erreichung des übergeordneten Ziels
  - UND-Knoten: **alle** untergeordneten Ziele müssen erreicht werden
  - ODER-Knoten: **mind. eines** der untergeordneten Ziele muss erreicht werden
- Blatt: elementarer Angriffsschritt
- Pfad zwischen Blatt und Wurzel = detaillierte Beschreibung einer Variante des Angriffs

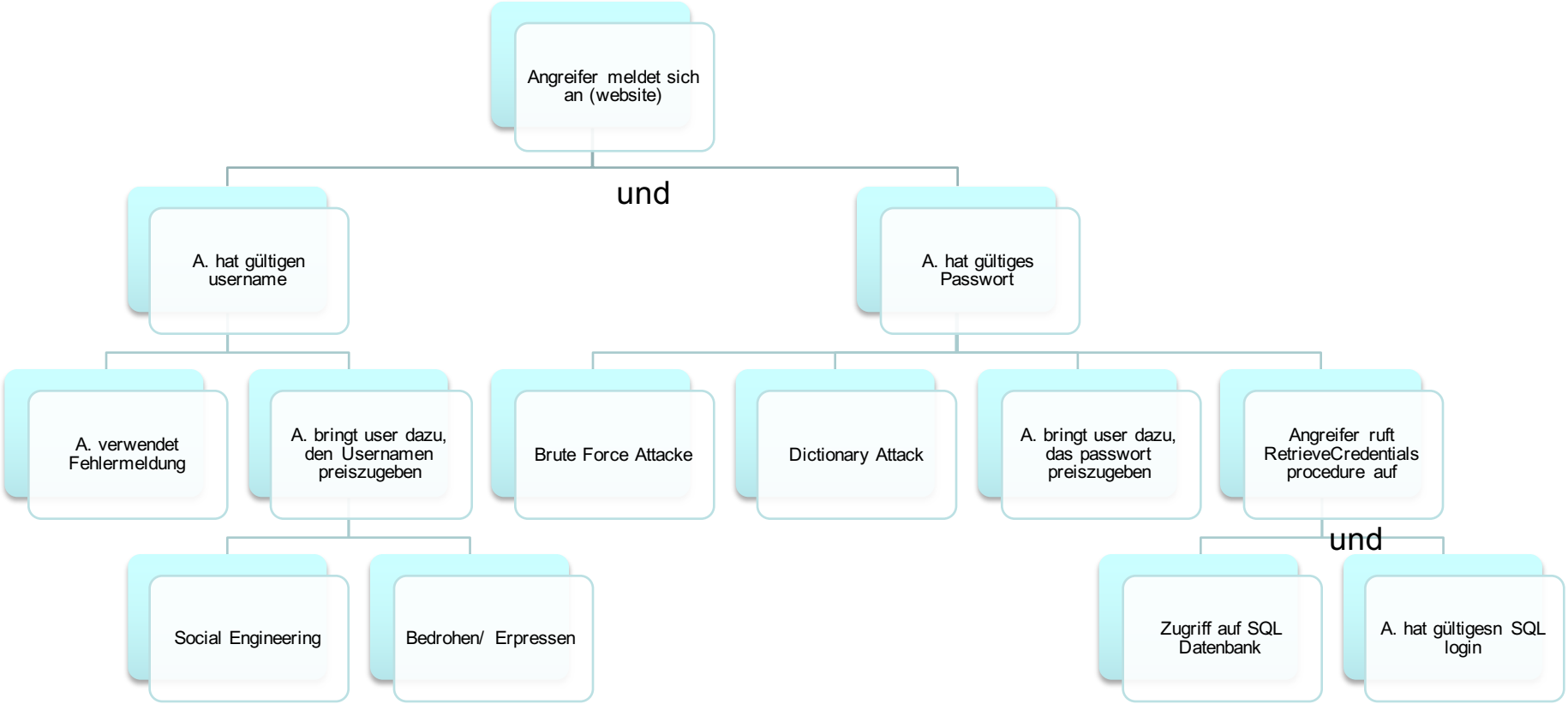
Bedrohungsbaum



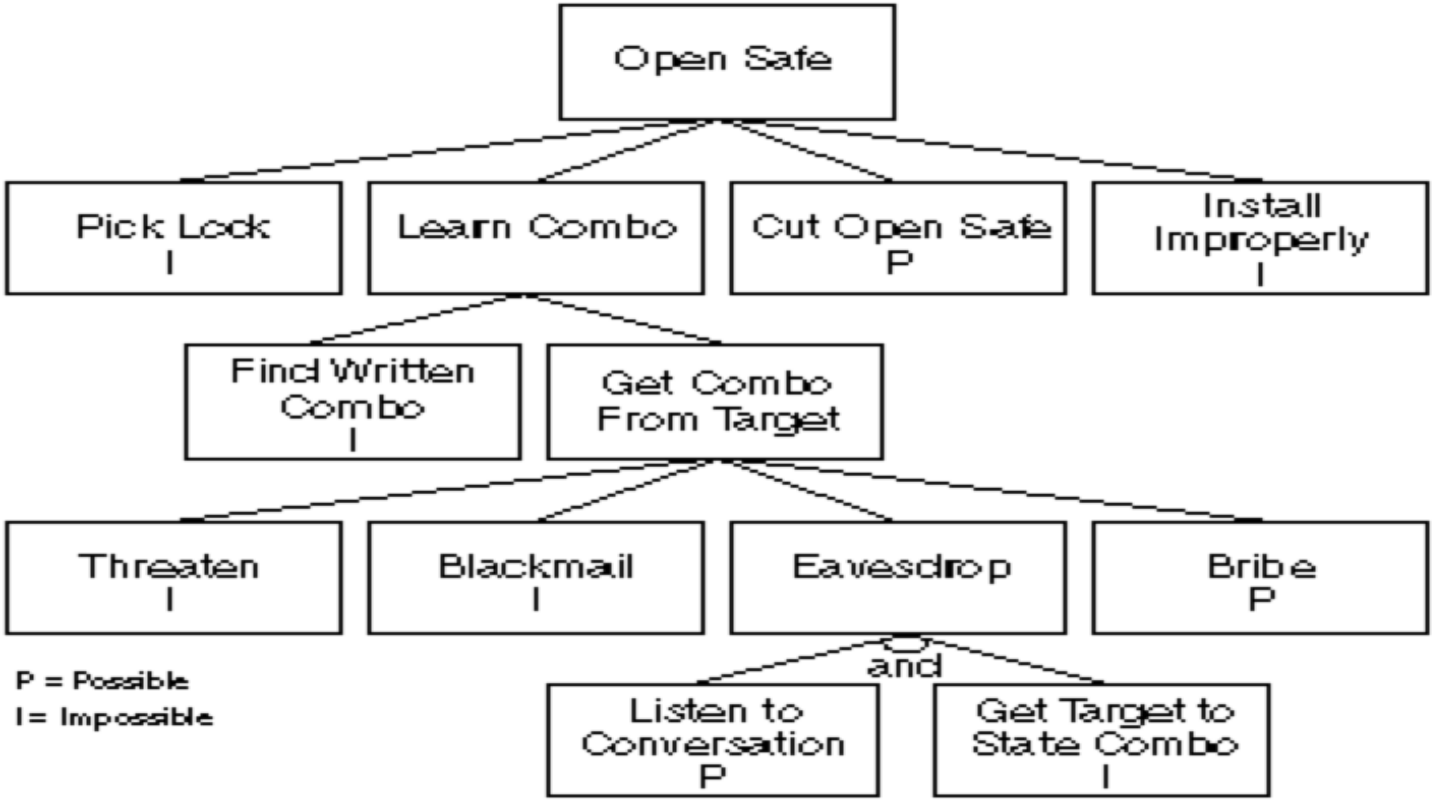
http://w3.inf.fu-berlin.de/lehre/WS05/SySi/

Klaus-Peter Löhr Karsten Otto: Systemsicherheit, 2012

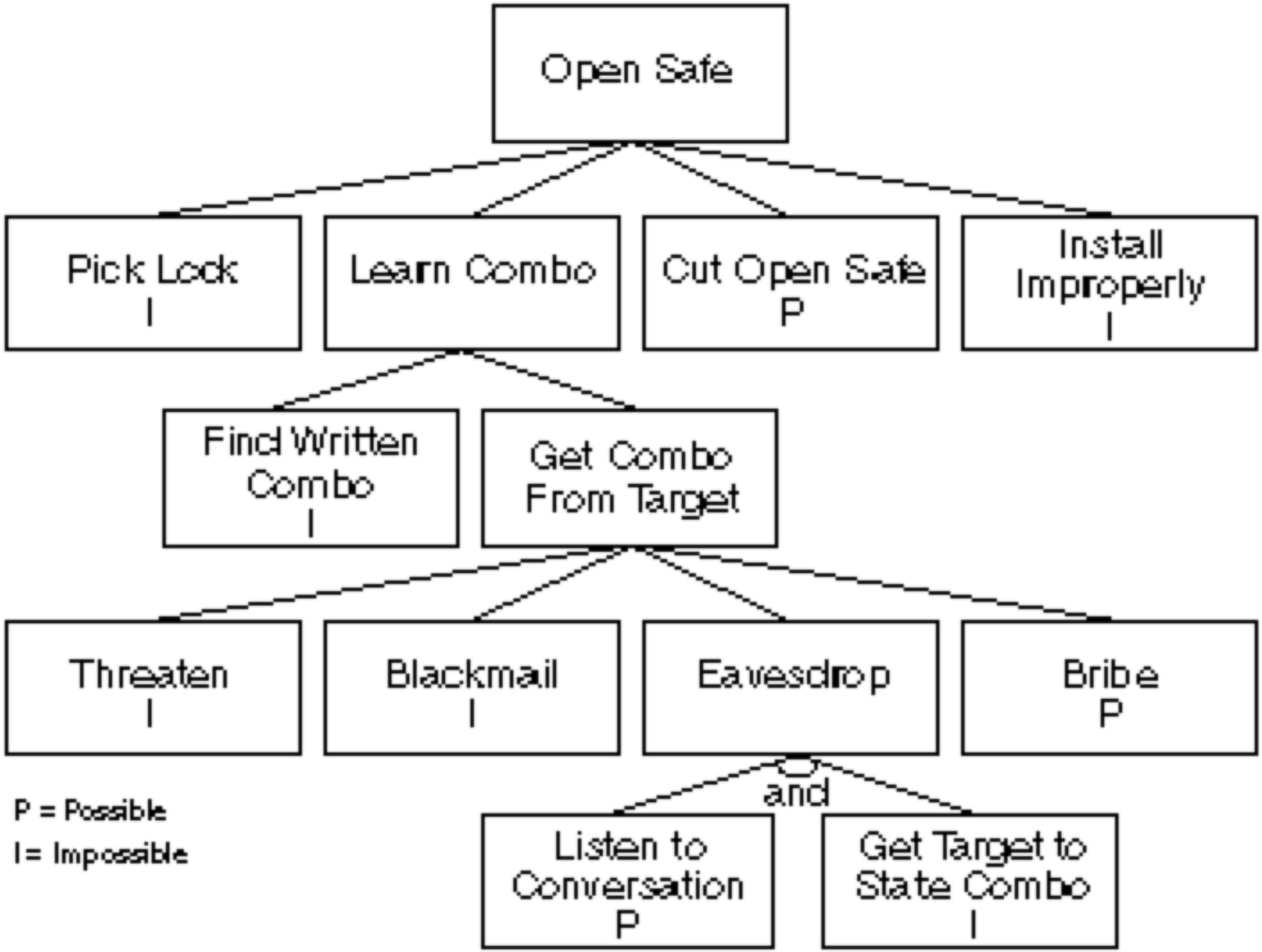
Bedrohungsanalyse: **Bedrohungsbaum**



Bedrohungsanalyse: **Bedrohungsbaum**



- Bruce Schneier: Attack Trees *Dr. Dobb's Journal* December 1999


Open Safe
Pick Lock
I
Learn Combo
Cut Open Safe
P
Install
Improperly
I
Find Written
Combo
I
Get Combo
From Target
Threaten
I
Blackmail
I
Eavesdrop
Bribe
P
and
Listen to
Conversation
P
Get Target to
State Combo
I
P = Possible
I = Impossible

Bedrohungsanalyse: **Bedrohungsbaum**

Bedrohungsbaum textuell

Ziel: Geldautomaten knacken

ODER 1. ohne Konto:

- 1.1 ODER Automat aufbrechen
- 1.2 ODER Automat mitnehmen

ODER 2. fremdes Konto:

- 2.1 ODER echte Karte:
  - 2.1.1 UND entwende Karte
  - 2.1.2 UND ermittle PIN
- 2.2 ODER falsche Karte:
  - 2.2.1 UND gefälschte Karte
  - 2.2.2 UND ermittle PIN

3. ODER eigenes Konto:

- 3.1 UND abheben
- 3.2 UND Abhebung bestreiten

Bedrohungsmodellierung: **Angriffsbaum**

- Ursprünglich stammt Angriffsbaum aus Umfeld HW-Produktion
  - zur Bestimmung von Fehlerwahrscheinlichkeiten
  - für jedes Blatt wird die Wahrscheinlichkeit des Auftretens bestimmt
  - mithilfe Wahrscheinlichkeitsrechnung kann bis zur Wurzel Fehlerwahrscheinlichkeit ermittelt werden

- Im Sec-SW-Engineering-Umfeld **kann** diese Bewertung gemacht werden, bringt aber wenig, da
  - Wahrscheinlichkeit eines Angriffs sehr schwer zu schätzen ist
    - (Motivation des Angreifers, Schwierigkeit des Angriffs…??)
  - Fehler in der Berechnung potenzieren sich nach oben!

- Sinn des Angriffsbaum für SW-Engineering:
  - Top-Level-Angriff wurde identifiziert
  - Aufteilung in detaillierte Angriffe, die besser auf die Architektur (“ports” etc.) gemappt werden können

Bedrohungsanalyse: **Bedrohungsmatrix**

- Klassifikation der Gefährdunsbereiche

= Zeilen der Matrix

  - HW-Angriff (z.B. Diebstahl von Speichermedien)
  - SW-Angriff (z.B. über Pufferüberlauf)
  - Bedrohung der Verfügbarkeit (Denial-of-Service)
  - Abstreiten (Repudiation)
  - Rechtemissbrauch

- Potentielle Auslöser von Bedrohungen

= Spalten der Matrix

  - Sysadmin
  - Programmierer
  - Benutzer (intern und extern)
  - Dienste, Protokolle
  - Mobiler Code

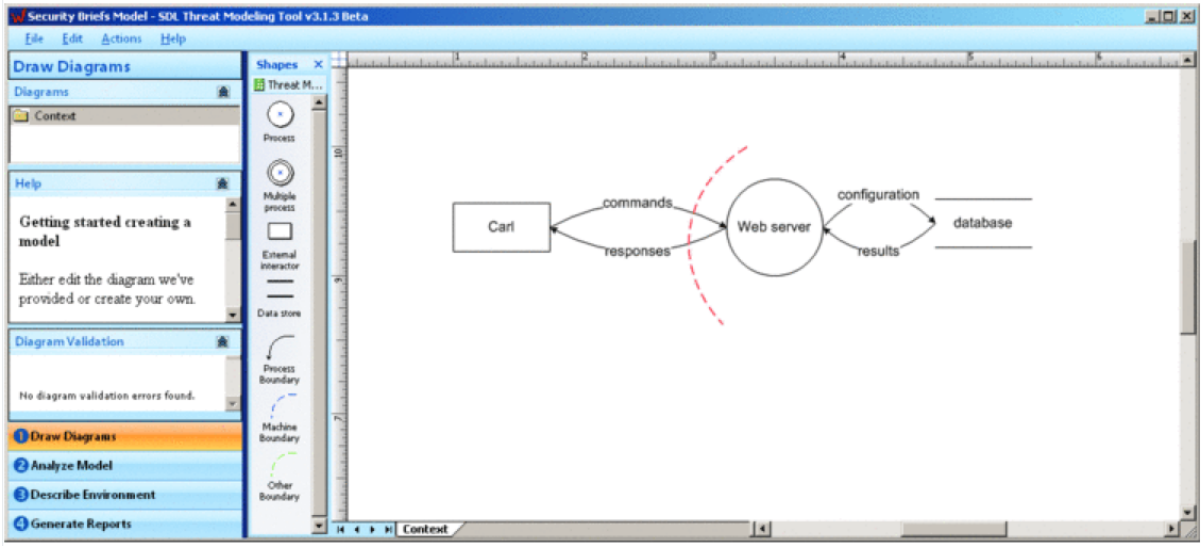
Claudia Eckert (2008): IT-Sicherheit, Oldenbourg

	Programmierer	interner Benutzer	externer Benutzer	mobiler Code
HW-Angriffe	-	Sicherungskopie vernichten	Einbruch ...	-
SW-Angriffe	Salami-Taktik	Dateischutz umgehen	Passwort knacken	Vireninfektion
Ressourcen-Blockade	Speicherverbrauch	Prozesse erzeugen	Netzlast erzeugen	Prozessor monopolisieren

Bedrohungsmodellierung: SW-Tools

- SDL Threat Modeling Tool von MS
  - nicht nur auf MS-Produkte anwendbar
- http://www.microsoft.com/security/sdl/adopt/threatmodeling.aspx

Gratis download



Quelle: http://msdn.microsoft.com/en-us/magazine/dd347831.aspx

Bedrohungsmodellierung: Phasen

1. Werte (und Akteure) identifizieren
2. Überblick Architektur
3. System zerteilen
4. Bedrohungen identifizieren
5. **Bedrohungen dokumentieren**
6. Bedrohungen bewerten
7. Gegenmaßnahmen planen

Bedrohungsmodellierung: Phasen

5. **Bedrohungen dokumentieren**
   - Standards (eigene oder fremde) verwenden
   - Informationen:
     - Bedrohung
     - mögliche Angreifer
     - Schnittstelle, wo Angriff erfolgt
     - Entwickler, der Bedrohung ident. hat

Bedrohungsmodellierung: Phasen

6. **Bedrohungen bewerten**

- Da meist nicht alle Bedrohungen behandelt werden können (Kosten, Zeit):
- risikobasierte Vorgehensweise notwendig

- Problem: Eintrittswahrscheinlichkeit und Schadenshöhe meist sehr schwer abschätzbar.
- Wenn Zahlen vorhanden (z.b. von sehr ähnlicher SW/ Prozess/ Branche):
  - risikobasierte Sicherheitsanalyse mit diesen Zahlen

- Weitere Methoden der Bewertung:
  - DREAD
  - OWASP Risk Rating

Risikoanalyse

Relevanz der einzelnen Bedrohungen beachten!

Bewertung der Risiken

- Wahrscheinlichkeit des Angriffs
- Schaden (geschätzter, potentieller) des Angriffs

Wahrscheinlichkeit hängt ab von:

- Aufwand zur Durchführung des Angriffs
- Nutzen des Angriffs für den Angreifer
- Angreifertyp (Risikobereitschaft, Motivation, Fähigkeiten, ...)

Risikoanalyse

- Systematische Risikoanalyse durch **Attributierung** des Bedrohungsbaums:
- Blätter werden mit diskreten oder kontinuierlichen Risiko-Attributen versehen, wie z.B. möglich/unmöglich oder z.b. grün, gelb, rot.
- Das Risiko-Attribut eines Nicht-Blatts kann rekursiv aus den Risiko-Attributen der Nachfolgerknoten berechnet werden,
- und somit kann das Risiko-Attribut der Wurzel ermittelt werden.

Bedrohungsmodellierung: DREAD

- von MS entwickelt
- weist einige Schwächen auf, daher von MS selbst nicht mehr verwendet
  - jedoch OWASP durch DREAD inspiriert

- ***Damage Potential***
  - If a threat exploit occurs, how much damage will be caused?
- ***Reproducibility***
  - How easy is it to reproduce the threat exploit?
- ***Exploitability***
  - What is needed to exploit this threat?
- ***Affected Users***
  - How many users will be affected?
- ***Discoverability***
  - How easy is it to discover this threat?

Bedrohungsmodellierung: DREAD

- ***Damage Potential***
  - If a threat exploit occurs, how much damage will be caused?
    - 0 = Nothing
    - 5 = Individual user data is compromised or affected.
    - 10 = Complete system or data destruction
- ***Reproducibility***
  - How easy is it to reproduce the threat exploit?
    - 0 = Very hard or impossible, even for administrators of the application.
    - 5 = One or two steps required, may need to be an authorized user.
    - 10 = Just a web browser and the address bar is sufficient, without authentication.
- ***Exploitability***
  - What is needed to exploit this threat?
    - 0 = Advanced programming and networking knowledge, with custom or advanced attack tools.
    - 5 = Malware exists on the Internet, or an exploit is easily performed, using available tools.
    - 10 = Just a web browser

Bedrohungsmodellierung: DREAD

- **Risk_DREAD** = (DAMAGE + REPRODUCIBILITY + EXPLOITABILITY + AFFECTED USERS + DISCOVERABILITY) / 5

= “Bedrohungslevel”

- Kritik an DREAD: Schaden und Eintrittswahrscheinlichkeit werden addiert
  - bei klassischer Risikoanalyse: Risiko = Schaden * Eintrittswahrscheinlichkeit
- Kritik2: Bewertungsschema für Schwachstellen und nicht Bedrohungen/Angriffen

Bedrohungsmodellierung: OWASP Risk Rating

- aus: “OWASP Testing Guide v4”
- https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

- 16 Faktoren strukturiert in 4 (2x2) Themengruppen

- 2 Themengruppen zu: **Wahrscheinlichkeit** eines erfolgreichen **Angriffs**
  - Threat Agent
  - Vulnerabiltiy
- 2 Themengruppen zu: **Schadensausmaß**
  - Technical Impact
  - Business Impact

Bedrohungsmodellierung: OWASP Risk Rating

Threat Agent Factors

- Welcher Angreifer? Falls verschiedene Angreifer möglich: verwende den “besten”.

- **Skill level**
  - How technically skilled? Security penetration skills (9), network and programming skills (6), advanced computer user (4), some technical skills (3), no technical skills (1)
- **Motive**
  - How motivated to find and exploit this vulnerability? Low or no reward (1), possible reward (4), high reward (9)
- **Opportunity**
  - What resources and opportunities are required to find and exploit this vulnerability? Full access or expensive resources required (0), special access or resources required (4), some access or resources required (7), no access or resources required (9)
- **Size**
  - How large is this group? Developers (2), system administrators (2), intranet users (4), partners (5), authenticated users (6), anonymous Internet users (9)

Bedrohungsmodellierung: OWASP Risk Rating

Vulnerability Factors

- Ziel: Wahrscheinlichkeit bestimmen.
- Verwende selber Angreifer von vorher

- **Ease of discovery**
  - How easy is it for this group of threat agents to discover this vulnerability? Practically impossible (1), difficult (3), easy (7), automated tools available (9)
- **Ease of exploit**
  - How easy is it for this group of threat agents to actually exploit this vulnerability? Theoretical (1), difficult (3), easy (5), automated tools available (9)
- **Awareness**
  - How well known is this vulnerability to this group of threat agents? Unknown (1), hidden (4), obvious (6), public knowledge (9)
- **Intrusion detection**
  - How likely is an exploit to be detected? Active detection in application (1), logged and reviewed (3), logged without review (8), not logged (9)

Bedrohungsmodellierung: OWASP Risk Rating

- **Technical Impact Factors**
  - Schadensausmaß;

- **Loss of confidentiality**
- How much data could be disclosed and how sensitive is it? Minimal non-sensitive data disclosed (2), minimal critical data disclosed (6), extensive non-sensitive data disclosed (6), extensive critical data disclosed (7), all data disclosed (9)
- **Loss of integrity**
- How much data could be corrupted and how damaged is it? Minimal slightly corrupt data (1), minimal seriously corrupt data (3), extensive slightly corrupt data (5), extensive seriously corrupt data (7), all data totally corrupt (9)
- **Loss of availability**
- How much service could be lost and how vital is it? Minimal secondary services interrupted (1), minimal primary services interrupted (5), extensive secondary services interrupted (5), extensive primary services interrupted (7), all services completely lost (9)
- **Loss of accountability**
- Are the threat agents' actions traceable to an individual? Fully traceable (1), possibly traceable (7), completely anonymous (9)

Bedrohungsmodellierung: OWASP Risk Rating

- **Business Impact Factors**

- **Financial damage**
- How much financial damage will result from an exploit? Less than the cost to fix the vulnerability (1), minor effect on annual profit (3), significant effect on annual profit (7), bankruptcy (9)
- **Reputation damage**
- Would an exploit result in reputation damage that would harm the business? Minimal damage (1), Loss of major accounts (4), loss of goodwill (5), brand damage (9)
- **Non-compliance**
- How much exposure does non-compliance introduce? Minor violation (2), clear violation (5), high profile violation (7)
- **Privacy violation**
- How much personally identifiable information could be disclosed? One individual (3), hundreds of people (5), thousands of people (7), millions of people (9)

Bedrohungsmodellierung: OWASP Risk Rating

Ermittlung der Werte:

- 2 Themengruppen zu: **Wahrscheinlichkeit** eines erfolgreichen **Angriffs**
  - Threat Agent
  - Vulnerabiltiy
  - **Mittelwert dieser 8 Faktoren**
- 2 Themengruppen zu: **Schadensausmaß**
  - Technical Impact
  - Business Impact
  - **Mittelwert dieser 8 Faktoren**
  - **oder nur Mittelwert des Business Impact**

Bedrohungsmodellierung: OWASP Risk Rating

- In Folge werden statt quant. Werte qualitative verwendet:

0 bis < 3	Niedrig
3 bis <6	Mittel
6 bis 9	Hoch

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

Bedrohungsmodellierung: Gegenmaßnahmen planen

- Typische Fragen:
  - Warum ist der Angriff möglich? Problem? Ist es gewünschte Funktionalität, die von anderen ausgenutzt wird? logischer Fehler?
  - Welche Änderungen an der Architektur können das Problem lösen? -> andere Schwachstellen dadurch?
  - Gibt es mehrere Angriffe die (gleiche Schwachstelle haben und/oder) gemeinsam abgewehrt werden können
  - Neue Vertrauensgrenzen notwendig?

- Gegenmaßnahmen ändern oft den Architekturentwurf -> evtl. Schleife in den Prozess

Bedrohungsmodellierung: Gegenmaßnahmen planen

- **Ergebnisse** der Planung Gegenmaßnahmen:
  - Tabelle mit ident. und bewerteten Angriffen (Punkt 5 Bedrohungen dokumentieren):
  - nun Gegenmaßnahmen dazu
  - für jeden abzuwehrenden Angriff: neue Sicherheitsanforderungen
    - Falls Anforderung bereits existiert, aber Angriff (bisher) nicht abgewehrt wurde, stimmt etwas im Prozess nicht!

Bedrohungs-modellierung

- Andere (aehnliche) Vorgehensweise

- Quelle: Matthias Rohr : Sicherheit von Webanwendungen in der Praxis

Schritt	Beschreibung
Schritt 1: Festlegung des Untersuchungsgegenstandes	Im ersten Schritt wird der Untersuchungsgegenstand (Target of Evaluation, TOE) klar festgelegt und abgegrenzt. Gewöhnlich sollten Backendsysteme, Middleware und zugrundeliegende Plattformen ausgegrenzt und im Rahmen eines separaten Bedrohungsmodells analysiert werden. Bezogen auf den Untersuchungsgegenstand sollten die folgenden Informationen ermittelt und dokumentiert werden: – Anwendungsname und Version – Zweck („Business Objectives“, wozu dient die Anwendung?) – Scope der Analyse (z. B. Ausgrenzen bestimmter Komponenten oder Bedrohungsarten) – Relevante Stakeholder – Kritikalität und bekannte Risiken/Gefährdungen – Existierende Sicherheitsvorgaben – Bisherige Sicherheitsuntersuchungen – Verwendeter Technologiestack
Schritt 2: Anwendungs-Dekomposition	Im Rahmen der Dekomposition einer Anwendung (engl. Application Decomposition) werden die für die Bedrohungsanalyse relevanten Anwendungseigenschaften untersucht und dokumentiert. Hierzu dient insbesondere die Befragung von sogenannten Subject Matter Experts (SMEs) wie Entwicklern, Architekten oder Projektleitern. Wichtige Bestandteile sind die Identifikation und Dokumentation von: – Systemübersicht – Assets (insb. schutzwürdige Daten) – Trust Level (einzelner Systemkomponenten und Akteure) – Trust Boundarys (Vertrauensgrenzen) – Eintrittspunkte – Sicherheitsaspekte der Softwarearchitektur – Relevanten Datenflüsse (z. B. der Anmeldung) – Relevanten Anwendungsfälle
Schritt 3: Clusterung	Viele Anwendungen sind aus mehreren (technologisch oder fachlich) heterogenen Komponenten zusammengesetzt. Würden sie über ein gemeinsames Bedrohungsmodell abgebildet werden, so würde dies die Analyse erheblich erschweren und könnte die Ergebnisse verfälschen. Ein häufiges Beispiel hierfür bilden administrative Schnittstellen, die ein völlig anderes Bedrohungsprofil besitzen als die für sonstige Benutzer zugänglichen Anwendungsbereiche. In einem solchen Fall empfiehlt es sich, die Anwendung in homogene (technische und fachliche) Cluster zu unterteilen. Dies ist nicht immer notwendig und wenn, dann sind es an dieser Stelle meist nur zwei oder drei Cluster, die sich identifizieren lassen. Alternativ kann es zweckmäßig sein, anstelle einer Clusterung die Bedrohungsanalyse zu „re-scopen“ und auf mehrere Modelle aufzuteilen

Bedrohungs-modellierung

- Andere (aehnliche) Vorgehensweise

- Quelle: Matthias Rohr : Sicherheit von Webanwendungen in der Praxis

Schritt	Beschreibung
Schritt 4: Bedrohungs-Identifikation	Identifikation der für eine Anwendung relevanten Bedrohungen (bzw. potentieller oder bereits akuter Gefährdungen) anhand unterschiedlicher Verfahren (Abschn. 4.1.4). Darunter Abuse Cases, DFD-Analysen, Fragenkataloge, Threat Profiling. Diese lassen sich je nach Anwendung, Analyst und Organisation spezifisch zusammenstellen. Das Ziel dieses Schrittes besteht darin, alle relevanten Bedrohungen (bzw. potenziellen Gefährdungen) für eine betrachtete Anwendung (und innerhalb eines gewählten Scopes) zu identifizieren. Es lässt sich dabei nicht vermeiden, dass ein und dieselbe Bedrohung mittels unterschiedlicher Verfahren mehrfach identifiziert wird. Das ist aber kein Problem, da diese im nächsten Schritt aufgeräumt werden
Schritt 5: Bedrohungs-Revision	– *Konsolidierung*: Zusammenführung identischer oder ähnlicher Bedrohungen. – *Identifikation abschwächender Faktoren*: Diese können organisatorischer, technischer oder physischer Natur sein (z. B. eine existierende Firewall oder organisatorische Regelungen). – *Einbeziehung bestehender Vorbewertungen*: Möglich etwa in Form von „relevant“ und „nicht relevant“. Dabei lassen sich auch sogenannte Known-Issues berücksichtigen, also Bedrohungen, die in der Vergangenheit bereits entsprechend bewertet worden sind
Schritt 6: Bewertung (Übergang zur Risikoanalyse)	Die konsolidierten und vorbewerteten Bedrohungen lassen sich nun mit Hilfe von unterschiedlichen Verfahren bewerten (Abschn. 4.5), wodurch sie zu tatsächlichen Gefährdungen bzw. Risiken qualifiziert werden. Neben CWSS und Bug Bars zählt zu den einsetzbaren Verfahren im Besonderen die Risikobewertung, die die Überführung einer Bedrohungsanalyse in eine Risikoanalyse ermöglicht

Bedrohungs-modellierung

- Andere (aehnliche) Vorgehensweise

- Quelle: Matthias Rohr : Sicherheit von Webanwendungen in der Praxis

Schritt 7: Ableitung von Gegenmaßnahmen	Die nun bewerteten Bedrohungen lassen sich im nächsten Schritt um entsprechende Maßnahmen (bzw. Anforderungen) ergänzen, die aus einzelnen Bedrohungen direkt abgeleitet werden. Es empfiehlt sich, eine Kategorisierung dieser Maßnahmen im Hinblick darauf vorzunehmen, dass sie sich den für die Umsetzung zuständigen Personen leichter zuordnen lassen: – *Organisatorisch*: z. B. Definition von Prozessen, Erstellen von Richtlinien – *Implementierung*: z. B. Korrektur von Schwachstellen im Code – *Konfigurativ*: z. B. Änderung von Einstellungen im Webserver – *Architekturell*: z. B. Installation einer neuen Identity-Management-Lösung
Schritt 8: Umsetzungs-Verifikation	In diesem Schritt wird eine Verifikation der Umsetzung von Maßnahmen im Hinblick auf Vollständigkeit, Korrektheit sowie Angemessenheit durchgeführt Hierzu lassen sich für viele der zuvor definierten Maßnahmen (genauso wie für die Bedrohung selbst) entsprechende Testfälle (Security Testcases) ableiten. In der Regel sollten in einem Bedrohungsmodell nur einige Stichpunkte gegeben werden. Für eine genauere Formalisierung solcher Tests bietet sich die Verwendung von separat spezifizierten Basistestfällen an (siehe Beispiel für „Cross-Site Request Forgery“ in Abschn. 4.10.6)

Bedrohungs-modellierung

- Andere (aehnliche) Vorgehensweise

- Quelle: Matthias Rohr : Sicherheit von Webanwendungen in der Praxis

Schritt	Beschreibung
Schritt 9: Lessons Learned	Analyse und Diskussion der im Rahmen der Durchführung gewonnenen Erkenntnisse. Neben Verbesserungsmöglichkeiten in Bezug auf Planung, Abstimmung, Ausführung sowie Dokumentation zählt hierzu insbesondere die Identifikation von inhaltlichen Erkenntnissen, welche als Input für nachfolgende Bedrohungsanalysen relevant sein können (Bedrohungskataloge). Beispiele: – Vorbewertungen (known issues) – (Basis-)Sicherheitstestfälle – Bedrohungskataloge – Metriken – (Generische) Maßnahmen
Schritt 10: Aktualisierung	Aktualisierung des Bedrohungsmodells (bzw. des Bedrohungsverzeichnisses) nach einer festgelegten Zeit und z. B. in Folge von relevanten Änderungen an der Anwendung oder aufgrund geänderter Sicherheitsannahmen

- Sehr gut natuerlich auch OWASP
- https://www.owasp.org/index.php/Application_Threat_Modeling

Literatur

- Paulus, Sachar (2011): Basiswissen Sichere Software: Aus- und Weiterbildung zum ISSECO Certified Professional for Secure Software Engineering. Auflage: 1. Heidelberg, Neckar: Dpunkt Verlag.
- Matthias Rohr (2015) : Sicherheit von Webanwendungen in der Praxis, kes Verlag

- OWASP