Sichere Webanwendungen

Lerneinheit 1: Einführung

Prof. Dr. Christoph Karg

Studiengang Informatik
Hochschule Aalen

Sommersemester 2023

Gliederung

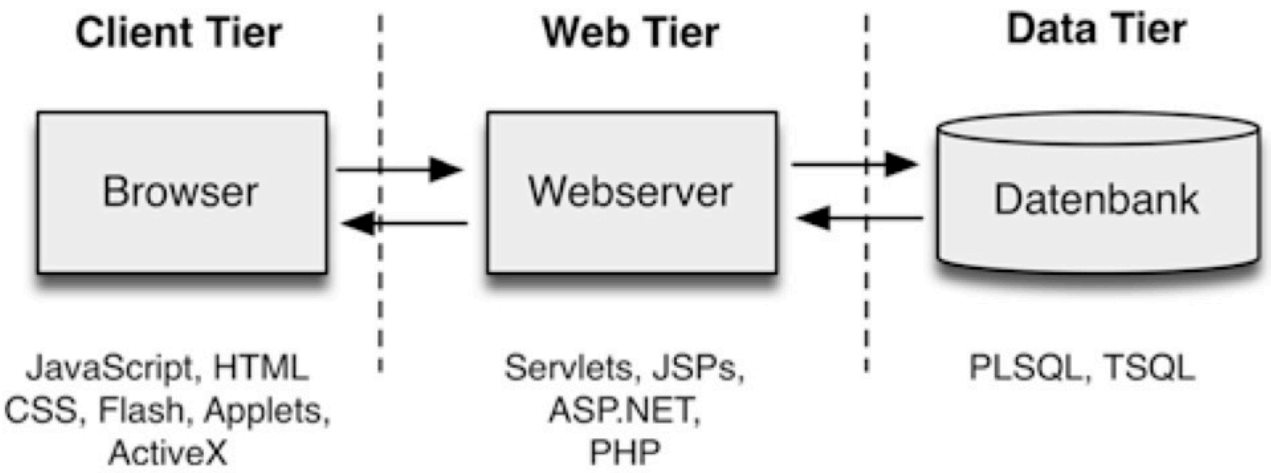
Begriff „Webanwendung“

- Eine Webanwendung ist eine auf Webtechnologien basierende Client-Server-Anwendung.
- Die Bedienung erfolgt über einen Webbrowser oder über eine dedizierte Client-Anwendung.
- Serverseitig ist die Webanwendung eine verteilte Anwendung bestehend aus Webserver, Applikationsserver und Datenbanksystem.
- Die Kommunikation zwischen Client und Server erfolgt über das HTTP- oder HTTPS-Protokoll.
- Bei der Entwicklung einer Webanwendung kommt in der Regel ein Webframework zum Einsatz.

3-Tier-Architektur

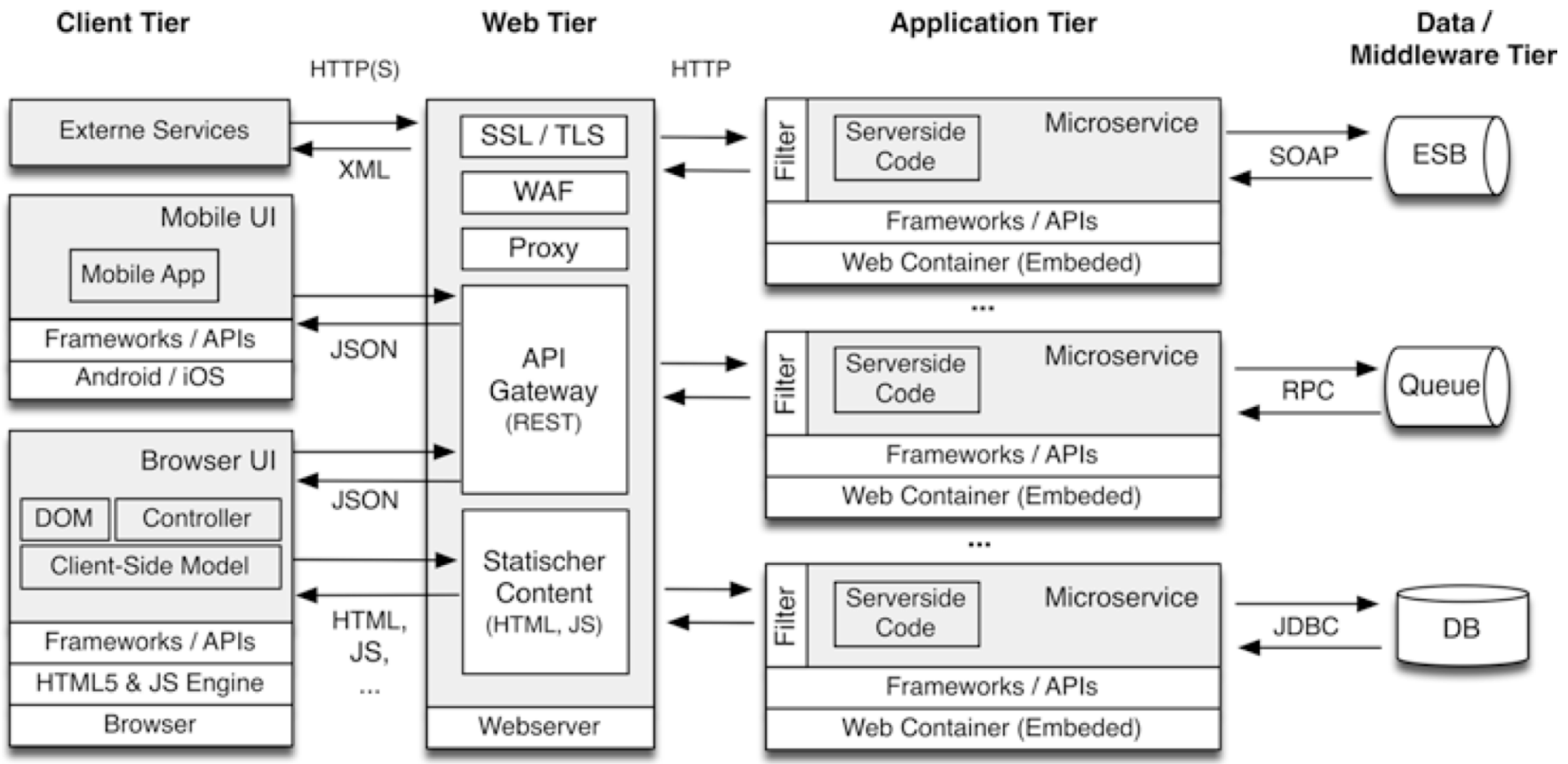
- Die 3-Tier-Architektur teilt eine Webanwendung in drei Schichten auf.
- Die Schichten sind:
  - Client Tier
  - Web Tier
  - Data Tier
- Bei modernen Webanwendungen ist diese Architektur deutlich differenzierter.
- Es kommt eine Vielzahl von Komponenten zum Einsatz, die zu einer Webanwendung zusammengefasst werden.

3-Tier-Architektur (einfach)



Quelle: [9]

3-Tier-Architektur (komplex)



Quelle: [9]

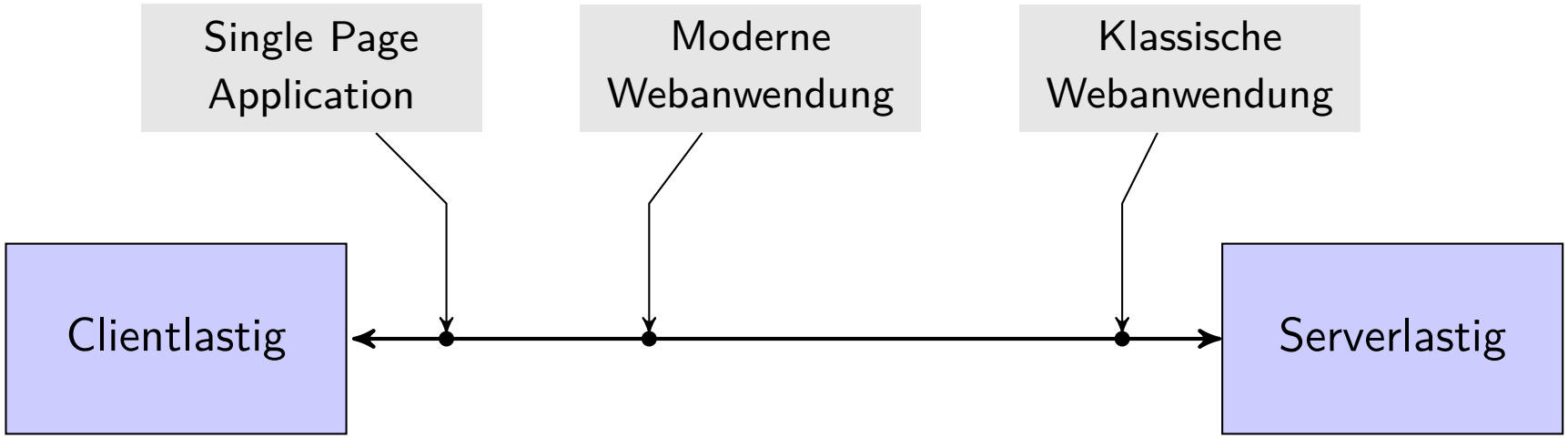
Software-Architektur

- Ein häufig vorzufindendes Enwurfsmuster bei Webanwendungen ist das Model View Controller (MVC) Pattern.
- Bestandteile:
  - Die Model Komponente beinhaltet die Daten der Anwendung.
  - Die View Komponente kümmert sich um die Darstellung der Daten.
  - Die Controller Komponente verwaltet die Model und View Komponenten.
- Die Auswahl des Entwurfsmusters ist für Sicherheitsmaßnahmen von zentraler Bedeutung.

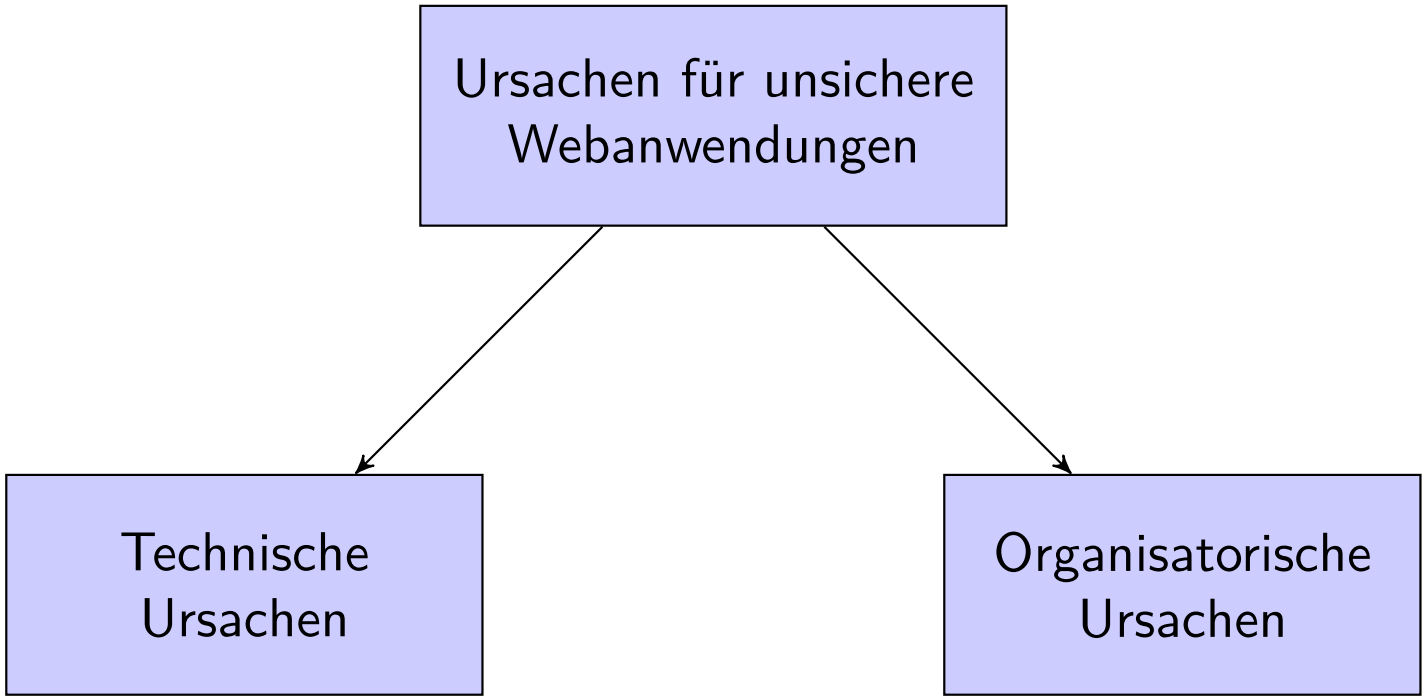
Ausführung des Codes

- Bei einer Webanwendung wird Code sowohl auf dem Server als auch auf dem Client ausgeführt.
- Bei „klassischen“ Webanwendungen werden die Seiten komplett auf dem Server berechnet und als HTML-Dokument ausgeliefert.
- In modernen Webanwendung erfolgt der Zugriff auf die Daten des Servers in vielen Fällen über eine REST-Schnittstelle.
- Sogenannte Single Page Applikationen werden hauptsächlich im Browser ausgeführt.

Verteilung der Code-Ausführung



Ursachen für unsichere Webanwendungen



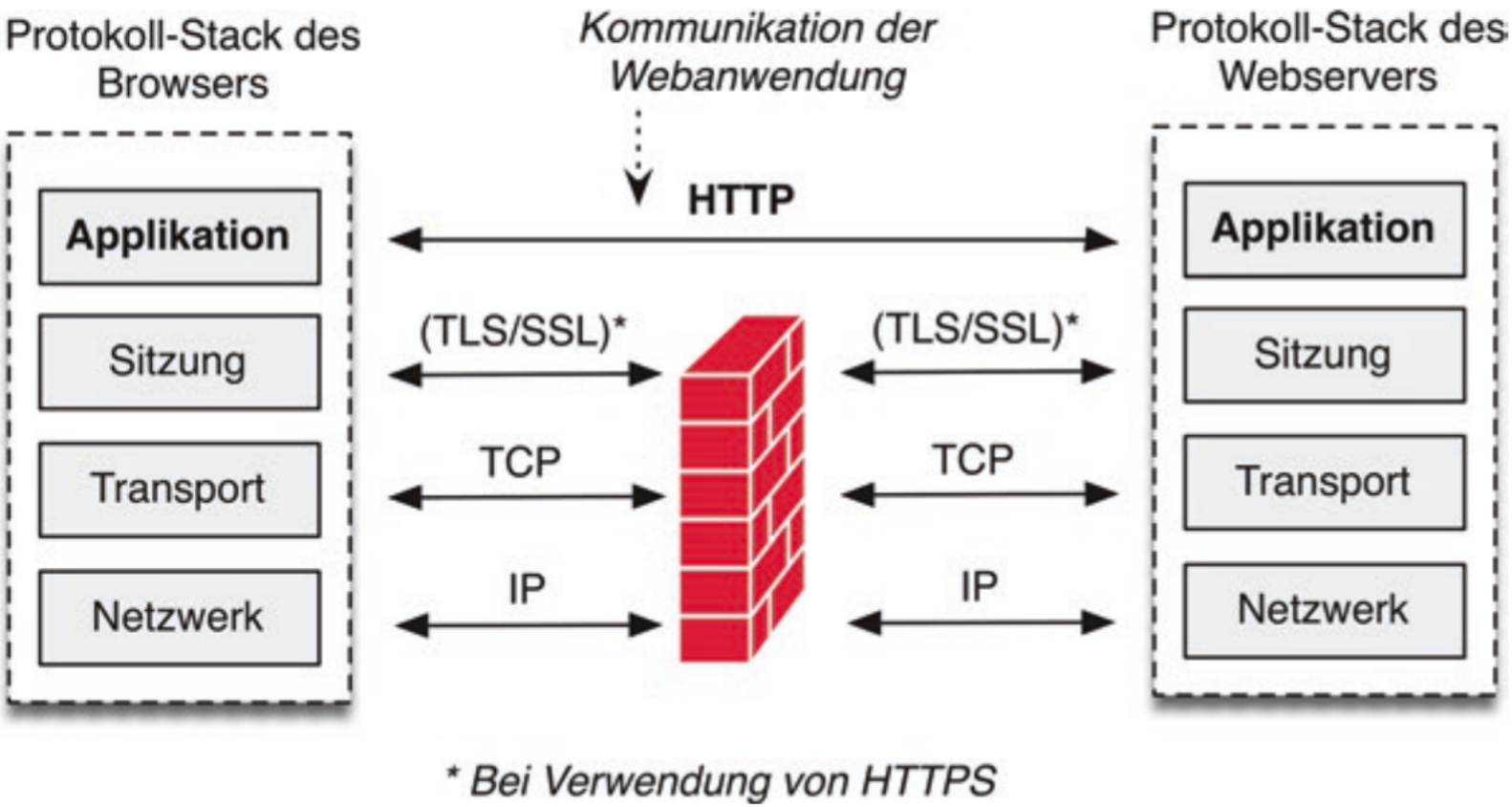
Technische Ursachen

- Anwendungs- versus Netzwerkschicht
- Design des HTTP-Protokolls
- Unzureichende Validierung der Eingabedaten
- Offenlegung serverseitiger Geschäftsfunktionen
- Enkodierungstechniken
- Clientseitige Ausführung von Code
- Unzureichende Absicherung der Backendsysteme

Anwendungs- versus Netzwerkschicht

- In vielen IT-Sicherheitsstandards liegt der Schwerpunkt auf der Absicherung der IT-Infrastruktur.
- Zur Absicherung von Netzwerken kommen häufig Firewalls und IDS-/IPS-Systeme zum Einsatz.
- Derartige Infrastruktur-Komponenten arbeiten überwiegend mit dem TCP/IP-Protokoll, also auf der Netzwerkebene,
- Angriffe auf höheren Schichten des Netzwerks wie z.B. der Anwendungsebene werden nur bedingt erkannt.
- Problem: die meisten Angriffe auf Webanwendungen finden auf der Anwendungsebene statt.

Kommunikation bei einer Webanwendung



Quelle: [9]

Design des HTTP-Protokolls

- Das HTTP-Protokoll wurde im Jahr 1996 von der IETF in der Version 1.0 standardisiert [1].
- Die aktuelle Version des HTTP-Protokolls ist 1.1 aus dem Jahr 2014 [6, 7, 5, 3, 4].
- Die Hauptziele beim Design von HTTP waren Interoperabilität und Robustheit.
- Beim Design von HTTP wurde kein Schwerpunkt auf Sicherheitsaspekte gelegt.
- Das Protokoll wurde nachträglich um SSL/TLS erweitert.

Sicherheitsaspekte von HTTP

- HTTP-Aufrufe kann man über die URL parametrisieren.
- HTTP besitzt keine Sicherheitsmechanismen für Vertraulichkeit und Datenintegrität.
- HTTP ist zustandslos.
- Die in HTTP enthaltenden Authentifizierungsverfahren sind unsicher.

Parametrisierung von HTTP-Aufrufen

- HTTP erlaubt die Übertragung von Parametern über die URL.
- Variante 1: Query String

  ```
  http://www.example.com/welcome.html
  ?param1=1&param2=2
  ```

  Da Query-Strings über die HTTP-GET-Methode übertragen werden, nennt man sie auch GET-Parameter.
- Variante 2: Nutzung des Pfad-Bereichs

  ```
  http://www.example.org/1/2/welcome.html
  ```

Alternative: POST-Methode

- Die POST-Methode ermöglicht die Übertragung von Parametern im Body des HTTP-Requests.
- Beispiel:

```
POST /welcome.html HTTP/1.1
Host: www.example.com
...
Content Length: 17
param1=1&param2=2
```

- In vielen Webframeworks werden sowohl die POST-Methode als auch die URL-Übertragung verwendet.

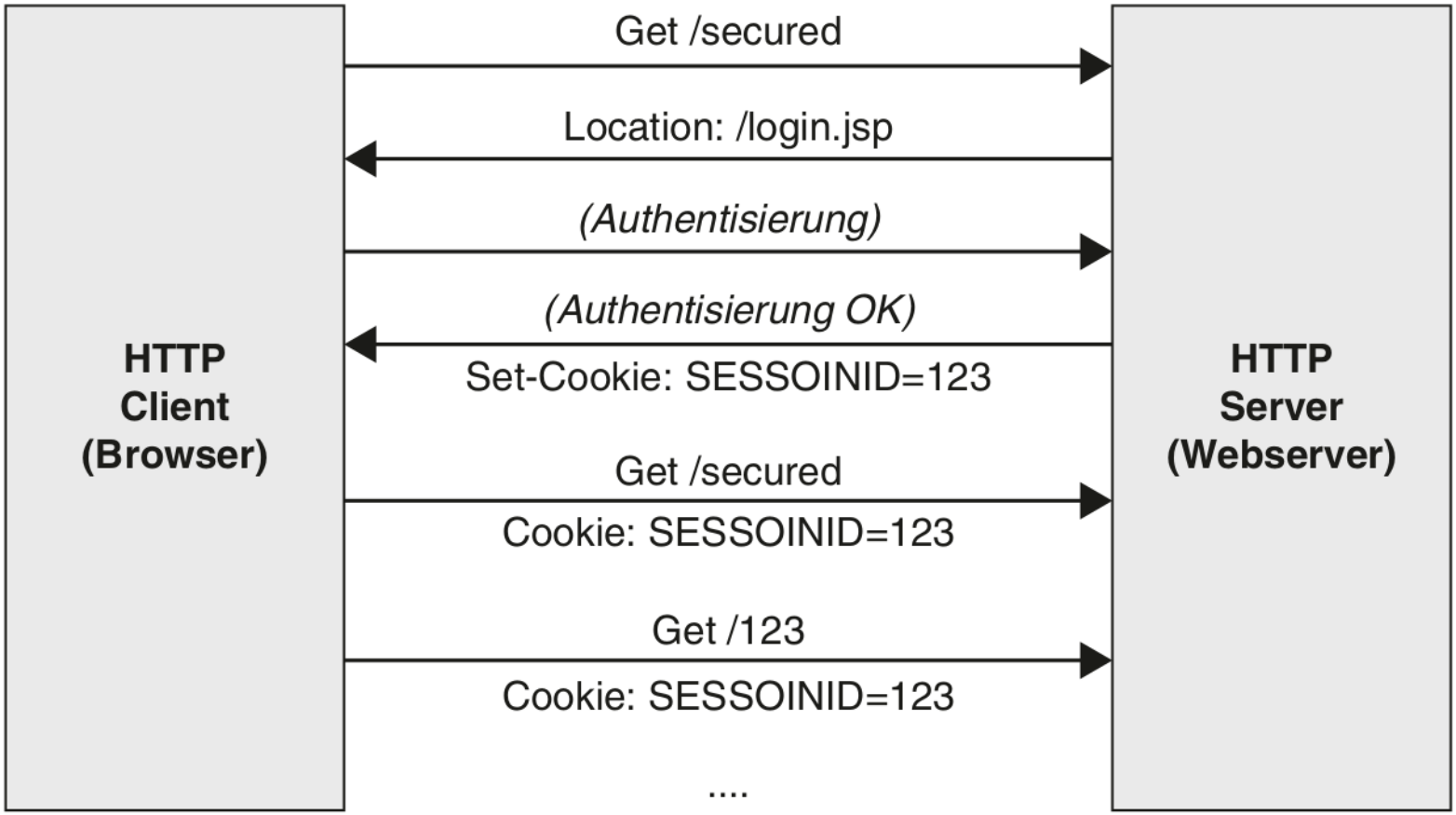
Fehlende Verschlüsselung und Datenintegrität

- HTTP enthält keine Mechanismen für die Verschlüsselung und die Gewährleistung der Integrität der übertragenen Daten.
- Dies ermöglicht das Abhören und die unautorisierte Manipulation der Kommunikation.
- HTTPS ist eine Erweiterung des Protokolls um eine zusätzliche SSL/TLS-Netzwerkschicht.
- HTTPS bietet Verschlüsselung, Schutz der Datenintegrität sowie Methoden zur Authentifizierung der Kommunikationspartner.
- Eine wichtige Rolle spielen in diesem Zusammenhang X.509-Zertifikate und Public-Key-Infrastrukturen.

HTTP ist zustandslos

- HTTP kennt keine Zustände, d.h., jeder HTTP-Request ist von jedem vorangegangenen unabhängig.
- Um serverseitig Informationen über mehrere HTTP-Requests aufrecht zu erhalten, ist ein Sitzungsmanagement erforderlich.
- Die Sitzung ist in der Regel an ein Benutzerkonto gekoppelt.
- Der Client verwendet zur Identifikation der Sitzung eine Session-ID (Cookie), die vom Server bei der Anmeldung des Nutzers vergeben wurde.
- Problem: Kennt der Angreifer die Session-ID, dann kann er die Sitzung übernehmen.

Cookie-basiertes Session Management



Quelle: [9]

Unsichere Authentifizierungsverfahren

- HTTP beinhaltet folgende Verfahren zur Authentifizierung von Benutzern:
  - HTTP Basic
  - HTTP Digest
- Jedes dieser Verfahren wird als unsicher eingestuft.
- Konsequenz: Oft implementieren die Entwickler einer Webanwendung selbst ein Verfahren zur Authentifizierung von Benutzern.
- Diese Eigenimplementierungen enthalten meistens gravierende Sicherheitsmängel.
- Besser: Einsatz einer Bibliothek zur Benutzerauthentifizierung.

Unzureichende Validierung der Eingabedaten

- Die in einer Webanwendung verarbeiteten Daten stammen häufig von HTTP-Parametern, die vom Benutzer eingegeben werden.
- HTTP-Parameter sind untypisiert, d.h., HTTP bietet keine Funktionalität für
  - die Festlegung des Datentyps eines Parameters und
  - die Korrektheit des übergebenen Wertes.
- Eine typische Art von Attacken auf Webanwendungen besteht in der Manipulation dieser Parameter.
- Konsequenz: Die Eingaben müssen vor der Verarbeitung von der Webanwendung auf Korrektheit überprüft werden.

Beispiel

- Eine Webanwendung verarbeitet folgende Parameter:

Name	*Wertebereich*
`gender`	$\{\texttt{male}, \texttt{female}\}$
`num`	$\{1, 2, 3, \ldots, 100\}$
`ok`	Boolean
`name`	Wort mit Groß- und Kleinbuchstaben

- Problem: In HTTP werden alle Parameter als String interpretiert.

Beispiel (Forts.)

- Korrekter Query String:

```
http://www.example.com/account?
gender=Male
&num=4
&ok=1
&name=Rudolf
```

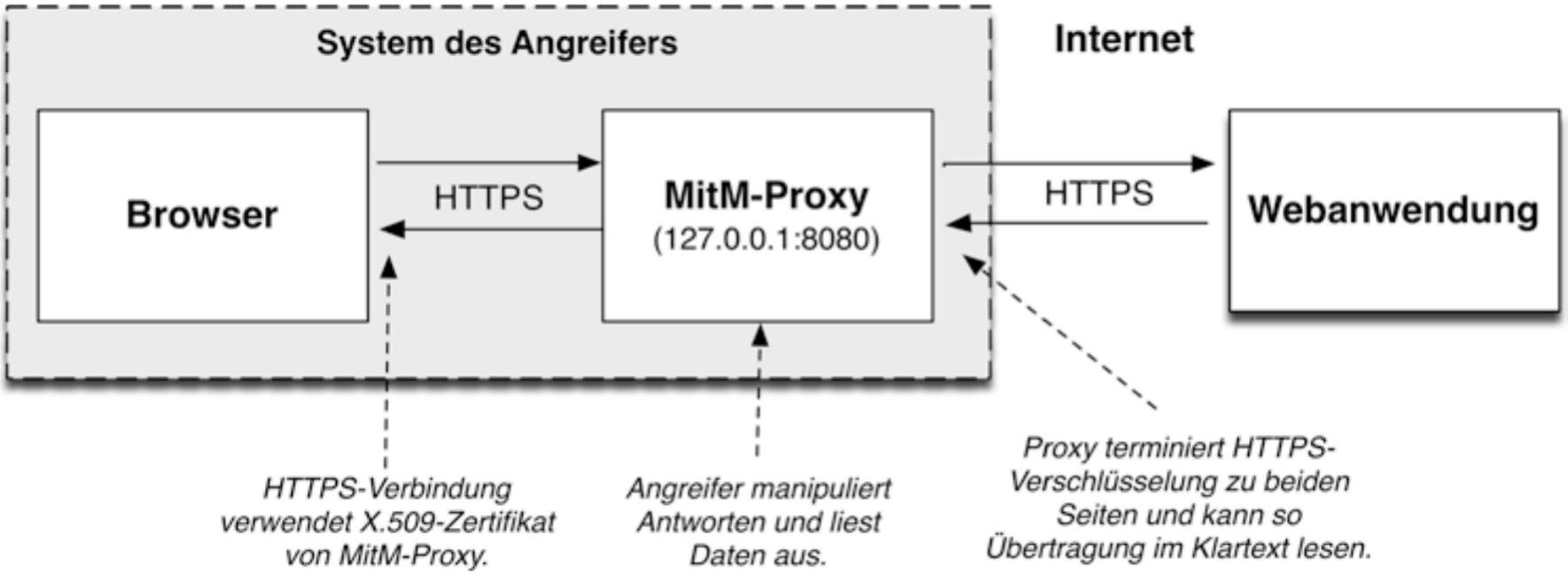
- Ebenfalls korrekt:

```
http://www.example.com/account?
gender="><script>exploit()</script>"
&num=-40
&ok=-20000
&name=Rudolf%00xxxxxxxxxxxxxxxx
```

Man-in-the-Middle (MITM) Angriffe

- Ein nicht zu verhindernder Angriffspunkt einer Webanwendung ist die Datenübertragung zwischen Client und Server.
- Bei einer Man-in-the-Middle (MITM) Attacke werden die vom Client an den Server gesendeten Daten gezielt manipuliert.
- MitM-Proxy ist ein Werkzeug, mit dem man diverse Manipulationen durchführen kann.
- Mit einem MitM-Proxy kann der Angreifer auch HTTPS-Verbindungen aufbrechen.

MitM-Proxy



Quelle: [9]

Offenlegung serverseitiger Geschäftsfunktionen

- Bei vielen Webanwendungen bildet ein Web Content Management System die Grundlage.
- Derartige Systeme besitzen administrative Schnittstellen, die ebenfalls webbasiert sind.
- Oft werden neben den Benutzereingaben auch interne Parameter (Anwendungsparameter) übertragen.
- Anwendungsparameter liefern Informationen über den internen Aufbau der Webanwendung.
- Durch die Manipulation der Anwendungsparameter lassen sich Angriffe durchführen.

Enkodierungstechniken

- Bei der Kodierung eines Query Strings wird zwischen normalen Zeichen und Steuerzeichen unterschieden.
- Anhand der Steuerzeichen (?, &, =, ...) wird beispielsweise ein Query String in seine Bestandteile zerlegt.
- Die Syntax von URIs wird in RFC 3986 spezifiziert [2].
- Ein über einen Query String übertragener Parameter kann serverseitig durch eine Komponente (SQL Query, Betriebsystem) weiterverarbeitet werden.
- Durch geschickte Wahl der Kodierung kann ein Angreifer seine Befehle in diese Komponenten einschleusen.

Clientseitige Ausführung von Code

- Bei vielen Webanwendungen wird Code im Browser ausgeführt, zum Beispiel Javascript.
- Die Entwickler der Webanwendung haben in der Regel keine Kontrolle über die Ausführungsumgebung.
- Mit den in viele Browser integrierten Entwicklungswerkzeugen kann man den Code analysieren und verändern.
- Im Code vorhandene Anwendungsparameter können gezielt manipuliert werden.

Unzureichende Absicherung der Backendsysteme

- Die Absicherung der Webanwendung ist nicht ausreichend für die Sicherheit des gesamten Systems.
- Nachgelagerte Komponenten wie Webserver, Datenbankmanagementsysteme oder Betriebssysteme müssen ebenfalls abgesichert werden.
- Veraltete Softwarebibliotheken mit Sicherheitslücken stellen ein ernstzunehmendes Risiko dar.
- Viele der im Internet zu erreichenden Webanwendungen werden nicht mehr weiterentwickelt oder regelmäßig gewartet.

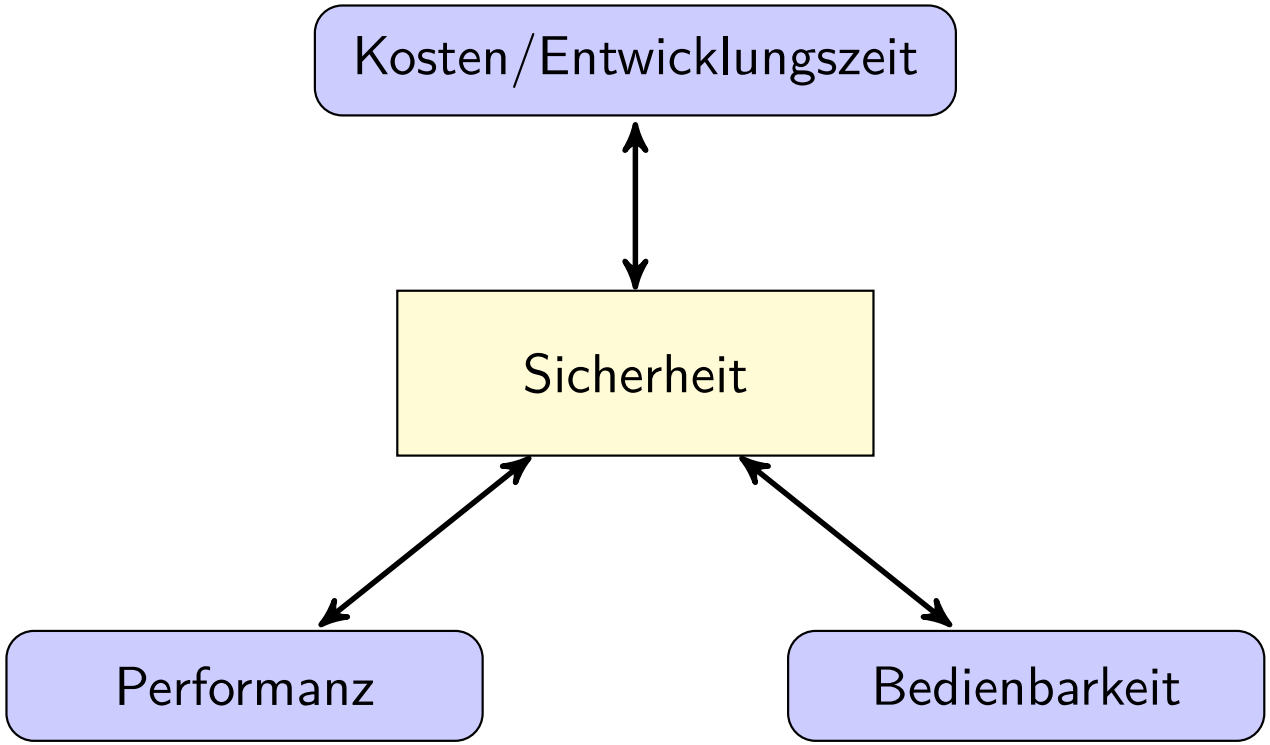
Organisatorische Ursachen

- Spannungsfeld zwischen Bedienbarkeit, Wirtschaftlichkeit und Sicherheit
- Fehlendes Fachwissen auf dem Gebiet der IT-Sicherheit
- Die Geschwindigkeit moderner Webentwicklung

Bedienbarkeit $\leftrightarrow$ Wirtschaftlichkeit $\leftrightarrow$ Sicherheit

- Der Zustand einer Webanwendung hinsichtlich ihrer Sicherheit lässt sich nur mit hohem Aufwand und entsprechendem Know-how bewerten.
- Der Einsatz von Sicherheitsmechanismen wirkt sich auf die Benutzbarkeit der Webanwendung aus.
- Tool-basierte Sicherheitsanalysen sind nicht ausreichend, um alle Sicherheitsmängel aufzuspüren.
- Die Umsetzung von Sicherheitsmaßnahmen resultiert in einem Aufwand, der von vielen Softwareentwicklern gescheut wird.
- Bei vielen Unternehmen ist das Bewusstsein für das Thema Sicherheit nur marginal ausgeprägt.

Zielkonflikt IT-Sicherheit



Quelle: [9]

Fehlendes Fachwissen bezüglich IT-Sicherheit

- Die von Schwachstellen in Webanwendungen ausgehenden Risiken werden oft unterschätzt.
- Vielen Software-Entwicklern fehlt das notwendige Verständnis für Sicherheitsbedrohungen und den erforderlichen Gegenmaßnahmen.
- Oft sind die Sicherheitsanforderungen für die Webanwendung ungenau oder gar nicht spezifiziert.
- Für viele Entwickler sind Sicherheitsaspekte bei der Software-Entwicklung nicht spannend und werden daher nicht berücksichtigt.

Die Geschwindigkeit moderner Webentwicklung

- Webanwendungen liegt ein agiler Softwareentwicklungsprozess zugrunde.
- Ein solcher Prozess besitzt viele Produktiterationen mit kurzen Releasezyklen.
- Ein agiles Vorgehen erschwert das Testen der Software und die Spezifikation und Gewährleistung von Sicherheitsanforderungen.
- Der Einsatz neuartiger Web-Technologien birgt die Gefahr neuer Sicherheitsrisiken.
- Eine überhastete Entscheidung für eine Technologie kann gravierende Auswirkungen auf die Sicherheit der Webanwendung haben.

Webanwendungssicherheit

- Webanwendungssicherheit ist eine Unterdisziplin der der IT-Sicherheit und Informationssicherheit.
- Webanwendungssicherheit befasst sich mit Sicherheitsaspekten von webbasierten Anwendungen sowie deren Komponenten und Diensten.
- Im Zentrum stehen Assets („wertvolle Zielobjekte“), denen ein konkreter Schutzbedarf zugeordnet werden kann.
- Der Schutzbedarf wird in anhand von Schutzzielen gemessen.
- Jede Sicherheitsmaßnahme erfüllt einen bestimmten Zweck, nämlich den Schutz von Assets.
- Das Sicherheitsniveau einer Webanwendung ist abhängig von ihrem Schutzbedarf sowie ihrer Erreichbarkeit.

Gängige Schutzziele

- Vertraulichkeit (Confidentiality)
  ⇝ Sensible Daten sind vor unbefugter Preisgabe geschützt.
- Integrität (Integrity)
  ⇝ Die Korrektheit (Unversehrtheit) von Daten und die korrekte Funktionsweise von Systemen ist sichergestellt.
- Verfügbarkeit (Availability)
  ⇝ Dienstleistungen, Funktionen eines IT-Systems, IT-Anwendungen oder IT-Netze können von den Anwendern stets wie vorgesehen genutzt werden.
- Authentizität (Authenticity)
  ⇝ Es ist gewährleistet, dass der Kommunikationspartner tatsächlich der ist, der er vorgibt zu sein, bzw., dass die Daten tatsächlich von der angegebenen Quelle stammen.

Ziel der Webanwendungssicherheit

These: Webanwendungssicherheit hat *nicht* zum Ziel, eine Webanwendung „sicher zu machen“.

Begründung:

- Man kann in der Regel nicht nachweisen, dass eine Webanwendung sicher ist.
- Der Begriff „sicher“ ist an sich nicht aussagekräftig: Sicher wovor? Wie sicher? Welche Art von Sicherheit?

Besser: Ziel der Webanwendungssicherheit ist es, bei einer Webanwendung ein gewisses Sicherheitsniveau herzustellen und durch entsprechende Testverfahren zu belegen.

Beachte: Eine Webanwendung kann ein gewisses Sicherheitsniveau besitzen und trotzdem unsicher sein.

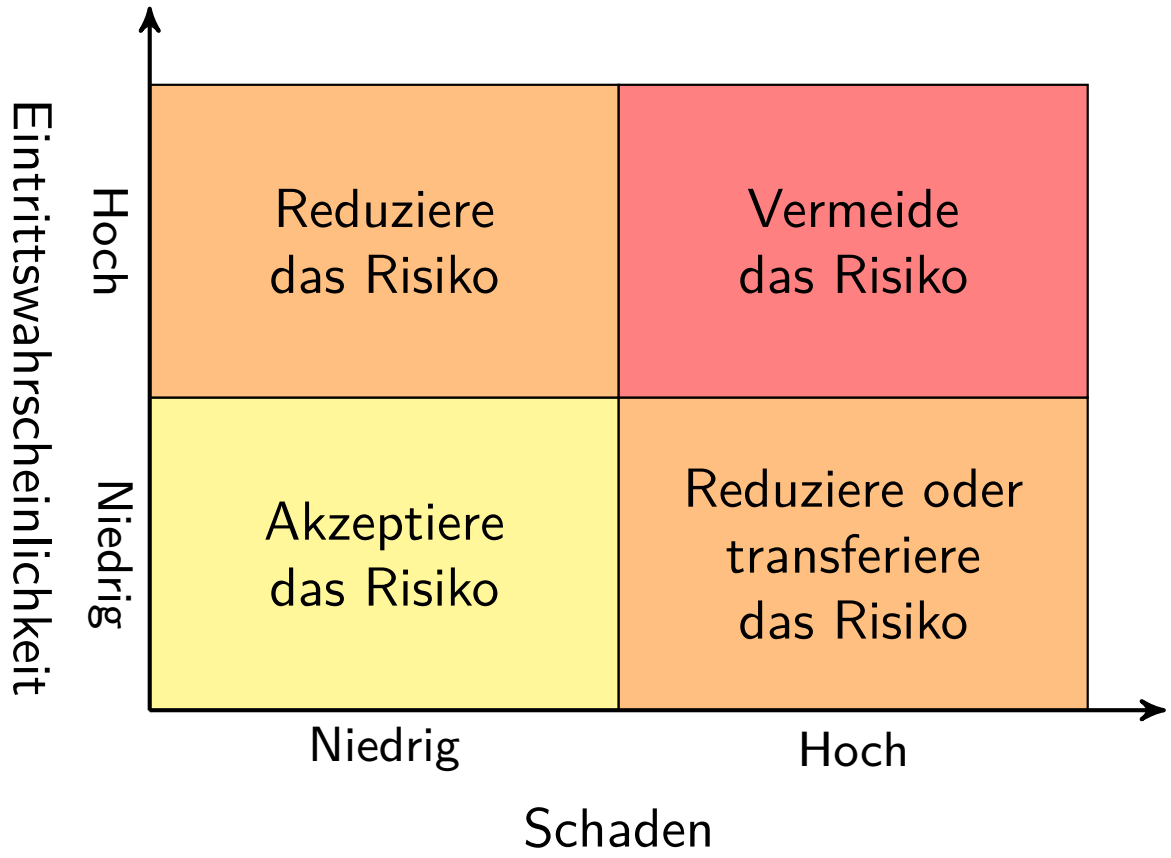
Sicherheitsrisiken

- Ein Sicherheitsrisiko ist eine mit einer Eintrittswahrscheinlichkeit und dem Schadenspotential bewertete Bedrohung, die durch die Ausnutzung einer Schwachstelle entsteht.
- Unterscheidung:
  - Risiken durch Schwachstellen technischer Art
  - Risiken durch Schwachstellen organisatorischer Art
  - Risiken durch Compliance-Verstöße, z.B., Verstöße gegen gesetzliche Vorschriften
- Webanwendungssicherheit beschäftigt sich mit der Identifikation und Reduktion potenzieller Risiken sowie der Erfüllung relevanter Sicherheitsanforderungen.

Management von Sicherheitsrisiken

- Um Sicherheitsrisiken zu identifizieren, muss der Anwendungskontext bekannt und verstanden sein.
- Die alleinige Existenz einer Schwachstelle in der Webanwendung bedeutet nicht automatisch ein hohes Risiko.
- Die Umsetzung einer technischen Sicherheitsmaßnahme ist eine Art der Risikoreduktion.
- Die Strategie zum Umgang mit Sicherheitsrisiken muss von den Eigentümern der Assets festgelegt werden.

Strategien zum Umgang mit Sicherheitsrisiken



Schutz von personenbezogenen Daten

- Webanwendungen verarbeiten in vielen Fällen personenbezogene Daten.
- Das Bundesdatenschutzgesetz (BDSG) und die EU-Datenschutzverordnung (EU-DSGVO) legen Anforderungen für die Sicherheit von personenbezogenen Daten fest.
- Ein Verstoß gegen diese Gesetze kann eine hohe Strafe nach sich ziehen.
- Um die Anforderungen umzusetzen, sollte der Datenschutzbeauftragte des Unternehmens in die Planung involviert sein.
- Werden Daten von Mitarbeitern verarbeitet, dann muss der Betriebsrat eingebunden werden bzw. existierende Betriebsvereinbarungen beachtet werden.

Prinzip der Vertrauenswürdigkeit

- Vertrauen spielt bei Entscheidungen der IT-Sicherheit eine wichtige Rolle.
- Unterscheidung:
  - Vertrauen in Personen und in Organisationen
  - Vertrauen in Software
  - Vertrauen in Systeme und Prozesse
- Zur Analyse von Vertrauensbeziehungen von Computersystemen werden Vertrauensgrenzen (Trust Boundaries) verwendet.
- Digitale Zertifikate auf Basis des X509-Standards spielen bei Webanwendungen eine wichtige Rolle, um die Echtheit von Servern zu bestätigen.

Aspekte der Softwarequalität

- Bei einer Webanwendung stellt Sicherheit genauso ein Qualitätsmerkmal dar wie Wartbarkeit oder Bedienbarkeit.
- Sicherheitsaspekte müssen in der Qualitätssicherung berücksichtigt werden.
- Die zu erreichenden Ziele werden anhand sicherheitsrelevanter Kriterien spezifiziert.
- Man unterscheidet Sicherheitsanforderungen funktionaler und nicht-funktionaler Art.
- Die Gewährleistung der Sicherheit einer Webanwendung muss über ihren gesamten Lebenszyklus erfolgen.

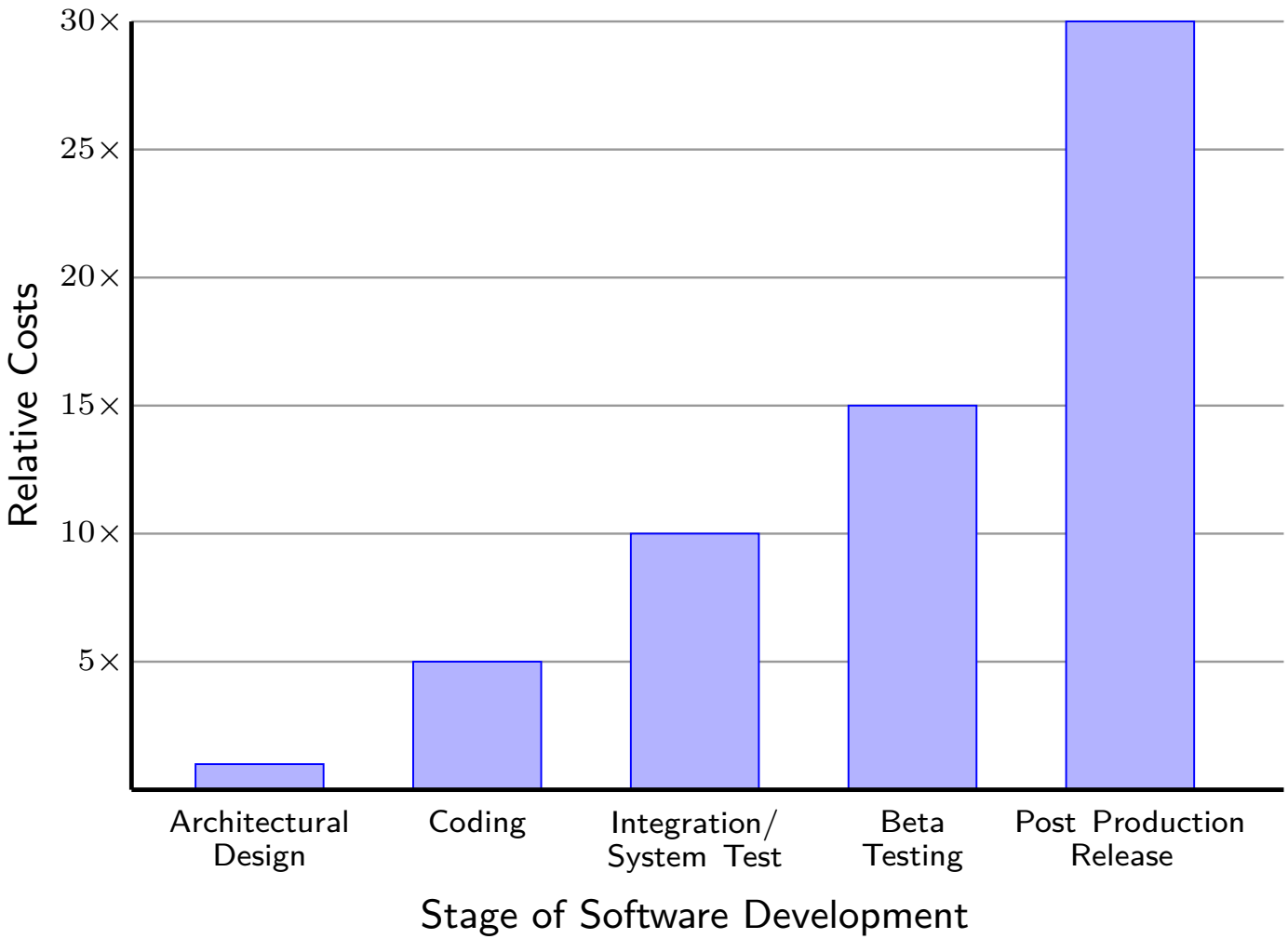
Phasen im Lebenszyklus einer (Web-)Anwendung

1. Konzeption
   - Spezifikation der Anwendung inklusive der Sicherheitsanforderungen
   - Auswahl passender Sicherheitsmechanismen
2. Implementierung
   - Implementierung der Anwendung
   - Testen der Implementierung
   - Abnahme der Anwendung
3. Betrieb
   - Produktiver Einsatz der Anwendung
   - Beseitigung von Schwachstellen und weiteren Bugs
   - Weiterentwicklung der Anwendung

Sicherheitsrelevante Qualitätskriterien

- Vertrauenswürdigkeit (Trustworthiness)
  $\rightsquigarrow$ Der Grad an Gewissheit, dass die Anwendung frei von Sicherheitsmängeln ist.
- Zurechenbarkeit (Dependability)
  $\rightsquigarrow$ Der Grad an Gewissheit, dass die Anwendung korrekt und wie vorhergesehen arbeitet.
- Resistenz (Survivability)
  $\rightsquigarrow$ Der Grad der Gewissheit, dass der Schaden im Falle einer Kompromittierung minimal ist und die Anwendung baldmöglichst wieder betriebsbereit ist.
- Konformität (Conformance)
  $\rightsquigarrow$ Die Anwendung erfüllt gängige Anforderungen, Standards und gesetzliche Vorgaben.

Relative Kosten für das Beheben von Bugs

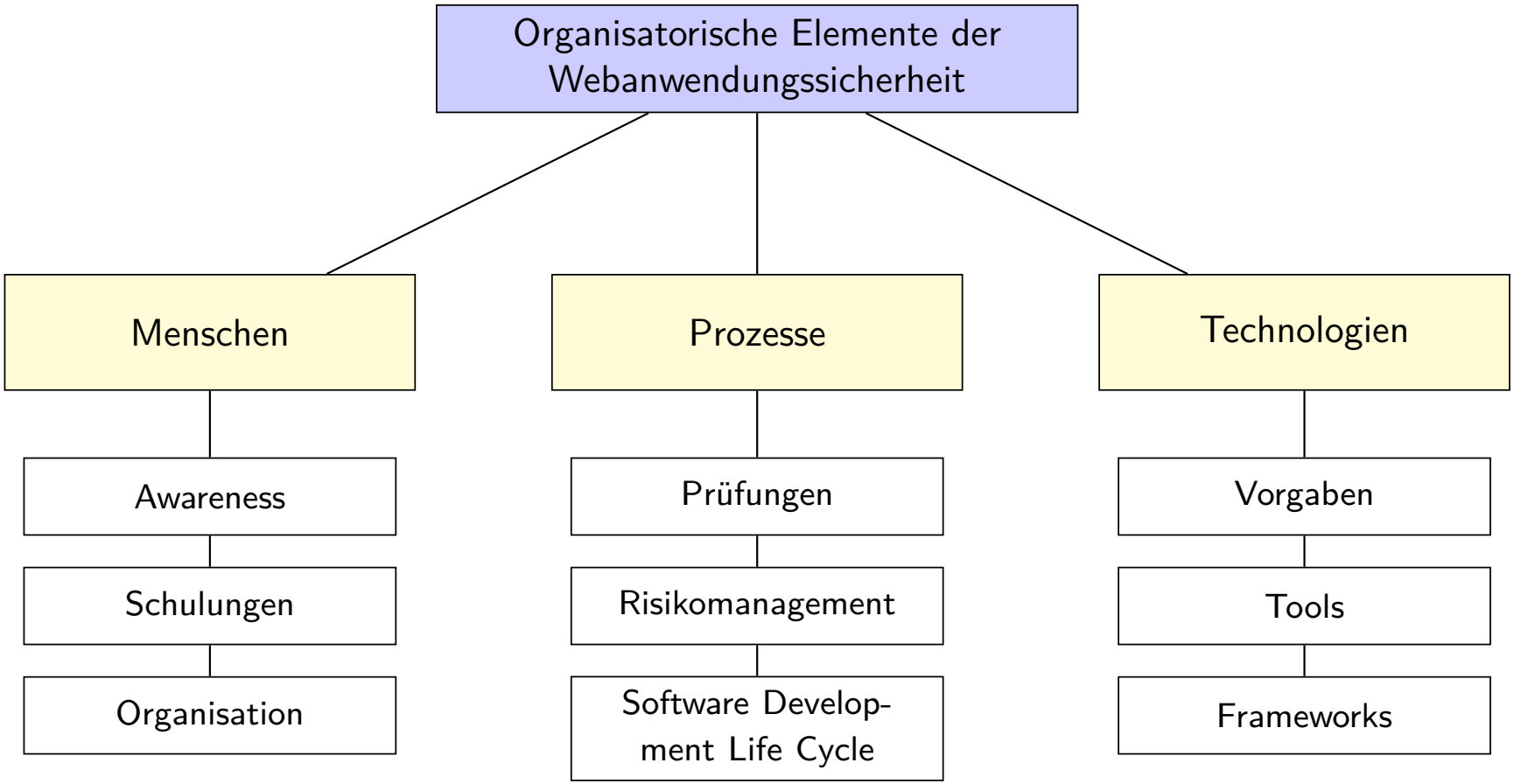


Quelle: [8]

Organisatorische Aspekte

- Webanwendungssicherheit ist in erster Linie ein organisatorisches Problem.
- Webanwendungssicherheit ist ein Querschnittsthema, das jede Phase des Lebenszyklus einer Webanwendung betrifft.
- Technische Maßnahmen sind in diesem Zusammenhang alleine nicht ausreichend.
- Vielmehr müssen entsprechende technische und organisatorische Prozesse in den Entwicklungsprozess integriert werden.
- Webanwendungssicherheit basiert auf den drei Säulen Menschen, Prozesse und Technologien.

Drei Säulen der Informationssicherheit



Zusammenfassung

- Eine Webanwendung ist eine komplexe verteilte Anwendung.
- Bei der Webentwicklung wird oft wenig Wert auf Sicherheitsaspekte gelegt.
- Ziel der Webanwendungssicherheit ist es, bei einer Webanwendung ein gewisses Sicherheitsniveau herzustellen.
- Die Sicherheit einer Webanwendung muss über deren gesamten Lebenszyklus gewährleistet werden.
- Webanwendungssicherheit ist in der ersten Linie ein organisatorisches Problem.

Literatur I

[1] Tim Berners-Lee, Roy Fielding und Henrik Frystyk. *Hypertext Transfer Protocol – HTTP/1.0*. Request for Comments 1945. Internet Engineering Task Force. 1996. URL: https://tools.ietf.org/html/rfc1945.

[2] Tim Berners-Lee, Roy Fielding und Larry Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. Request for Comments 3986. Internet Engineering Task Force. 2005. URL: https://tools.ietf.org/html/rfc3986.

[3] Roy Fielding, Yves Lafon und Julian Reschke, Hrsg. *Hypertext Transfer Protocol (HTTP/1.1): Range Requests*. Request for Comments 7233. Internet Engineering Task Force. 2014. URL: https://tools.ietf.org/html/rfc7233.

Literatur II

[4] Roy Fielding, Mark Nottingham und Julian Reschke, Hrsg. *Hypertext Transfer Protocol (HTTP/1.1): Caching*. Request for Comments 7234. Internet Engineering Task Force. 2014. URL: https://tools.ietf.org/html/rfc7234.

[5] Roy Fielding und Julian Reschke, Hrsg. *Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests*. Request for Comments 7232. Internet Engineering Task Force. 2014. URL: https://tools.ietf.org/html/rfc7232.

[6] Roy Fielding und Julian Reschke, Hrsg. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. Request for Comments 7230. Internet Engineering Task Force. 2014. URL: https://tools.ietf.org/html/rfc7230.

Literatur III

[7] Roy Fielding und Julian Reschke, Hrsg. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. Request for Comments 7231. Internet Engineering Task Force. 2014. URL: https://tools.ietf.org/html/rfc7231.

[8] NIST, Hrsg. *The Economic Impacts of Inadequate Infrastructure for Software Testing. Final Report*. National Institute of Standards und Technology. 2002. URL: https://www.nist.gov/sites/default/files/documents/director/planning/report02-3.pdf.

[9] Matthias Rohr. *Sicherheit von Webanwendungen in der Praxis*. 2. Aufl. Springer Vieweg, 2018.