Software Engineering und Projektmanagement: Sicherheit in der Softwareentwicklung

Florian Fankhauser, Christian Schanes
ESSE – Establishing Security

INSO – Industrial Software
Institut für Rechnergestützte Automation | Fakultät für Informatik | Technische Universität Wien

Agenda

IT-Sicherheit

- IT-Sicherheit ist viel zu umfangreich, um sie in ca. 90min zu lehren!
- Daher heute nur Vorstellung einiger wichtiger Ideen/Konzepte

- Weitere Vertiefung: ESSE Lehrveranstaltungen
  - Introduction to Security (183.594)
  - Security for Systems Engineering (183.637)
  - Advanced Security for Systems Engineering (183.645)
  - IT Security in Large IT Infrastructures (183.633)
  - Seminar aus Security (183.606)
  - Projekte, Bakkalaureatsarbeiten, Diplomarbeiten, Dissertationen

Motivation für IT-Sicherheit

- NSA-Skandal: Aktualisierte Zeitleiste zu den Snowden-Enthüllungen
- PGP-Erfinder zur NSA-Affäre: Sicherheit rechtfertigt keinen Polizeistaat
- Bash-Lücke: ShellShock ist noch nicht ausgestanden
- iOS 8 verrät Drittanbieter-Apps, mit wem man telefoniert
- Nikon D750 verteilt Fotos freizügig im WLAN
- Der GAU für Verschlüsselung im Web: Horror-Bug in OpenSSL

- Sicherheitslücken sind heute oft ein Geschäftsmodell
- Immer wieder treten (schwere) IT-Sicherheitslücken mit realen Folgen für Unternehmen und Personen auf!

Definition Sicherheit

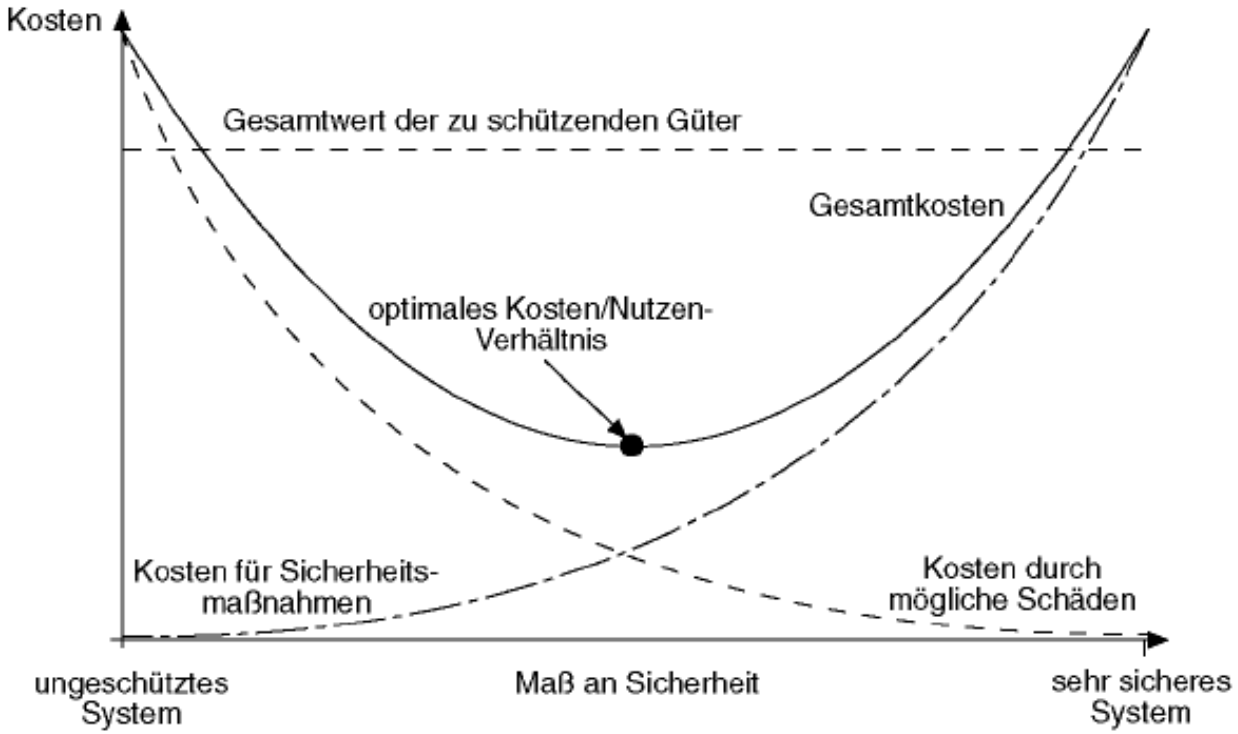
- Definition nach DIN VDE 31000
  - „Sicherheit ist eine Sachlage, bei der das Risiko nicht größer als das Grenzrisiko ist.“
  - „Das Grenzrisiko ist das größte noch vertretbare Risiko eines bestimmten technischen Vorganges oder Zustandes.“
  - „Eine absolute Sicherheit ohne jegliches Risiko gibt es weder in der Technik noch in der Natur.“
- Risiko = Schaden $*$ Eintrittswahrscheinlichkeit
- Das gilt auch für IT-Sicherheit!

Mehrere Phasen eines Angriffs

- Sammeln von Daten über das Angriffsziel
  - z.B. Verwendete Systeme, User-Kennungen,...
- Ausnutzen von gefundenden Sicherheitsproblemen (Zugriff, Ausweiten der Rechte)
  - z.B. SQL-Injection auf eine Datenbank
- Aufrechterhaltung des Zugriffs
  - z.B. Anlegen eines eigenen Benutzer-Accounts
- Verwischen von Spuren
  - z.B. Löschen von Logfiles

(Vergleiche E. Skoudis: *Counter Hack*)

Kosten-/Nutzenanalyse



(Vergleiche M. Raepple: Sicherheitskonzepte für das Internet)

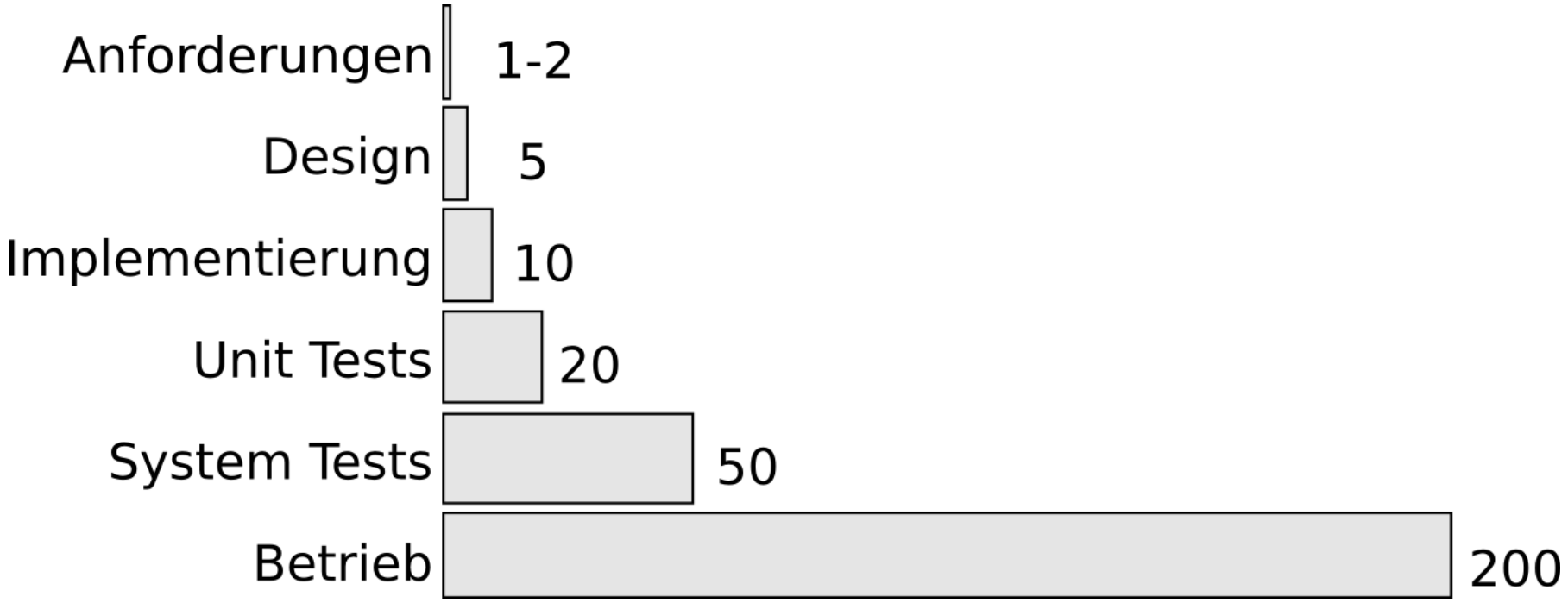
Softwareentwicklungsprozess – Phasen

Die weitere Betrachtung erfolgt auf Basis folgender Entwicklungsphasen:

- Analyse/Anforderungen
- Entwurf
- Implementierung
- Test
- Betrieb

Sicherheit ist kein Feature – „Sicherheit ist ein Prozess“ (Bruce Schneier)
$\rightarrow$ es ist nicht ausreichend Sicherheit in nur einem der Schritte zu betrachten (z.B. Verwendung einer Firewall). Sicherheit muss vom Management verstanden, getragen und gelebt werden!

Relative Kosten von Softwarefehlern



Z.B. wenn die Fehlerbehebung in der Phase der Anforderungen 1 EUR kostet, kann das 50 EUR in der Phase der System Tests kosten

(Vergleiche Stuart R. Faulk, 1995)

Sicherheit in der Analyse-/Anforderungsphase



INSO – Industrial Software

Institut für Rechnergestützte Automation | Fakultät für Informatik | Technische Universität Wien

Analyse

Als Analyse wird die Erhebung und Aufbereitung der Sicherheitsanforderungen für das zu entwickelnde System bezeichnet.

- Funktionale und nicht-funktionale Sicherheitsanforderungen
- Unterschiedliche/widersprüchliche Anforderungen von Stakeholdern (am System beteiligte Personen)
- Weitere Anforderungen durch Normen, Gesetze, Standards
- Bei sicherheitskritischen Systemen oft Evaluierungen:
  - Berücksichtigung der Evaluierungsanforderungen
  - Z.B. Evaluierungen anhand von Protection Profiles
- Sicherheit wird als selbstverständlich angesehen und nicht explizit von den Stakeholdern gefordert

Sicherheitsziele/Schutzziele

- Betrachtung unterschiedlicher Sicherheitsziele, i.A. v.a.
  - Vertraulichkeit
  - Integrität
  - Verfügbarkeit
  - → „*CIA Triad*“ (Confidentiality, Integrity, Availability)
- Weitere Sicherheitsziele sind z.B.
  - Authentizität
  - Nichtabstreitbarkeit

(Vergleiche BSI: IT-Grundschutz-Kataloge)

Schutzbedarf

Kategorien können z.B. sein:

- normal
- hoch
- sehr hoch

- gering
- mittel
- hoch
- sehr hoch

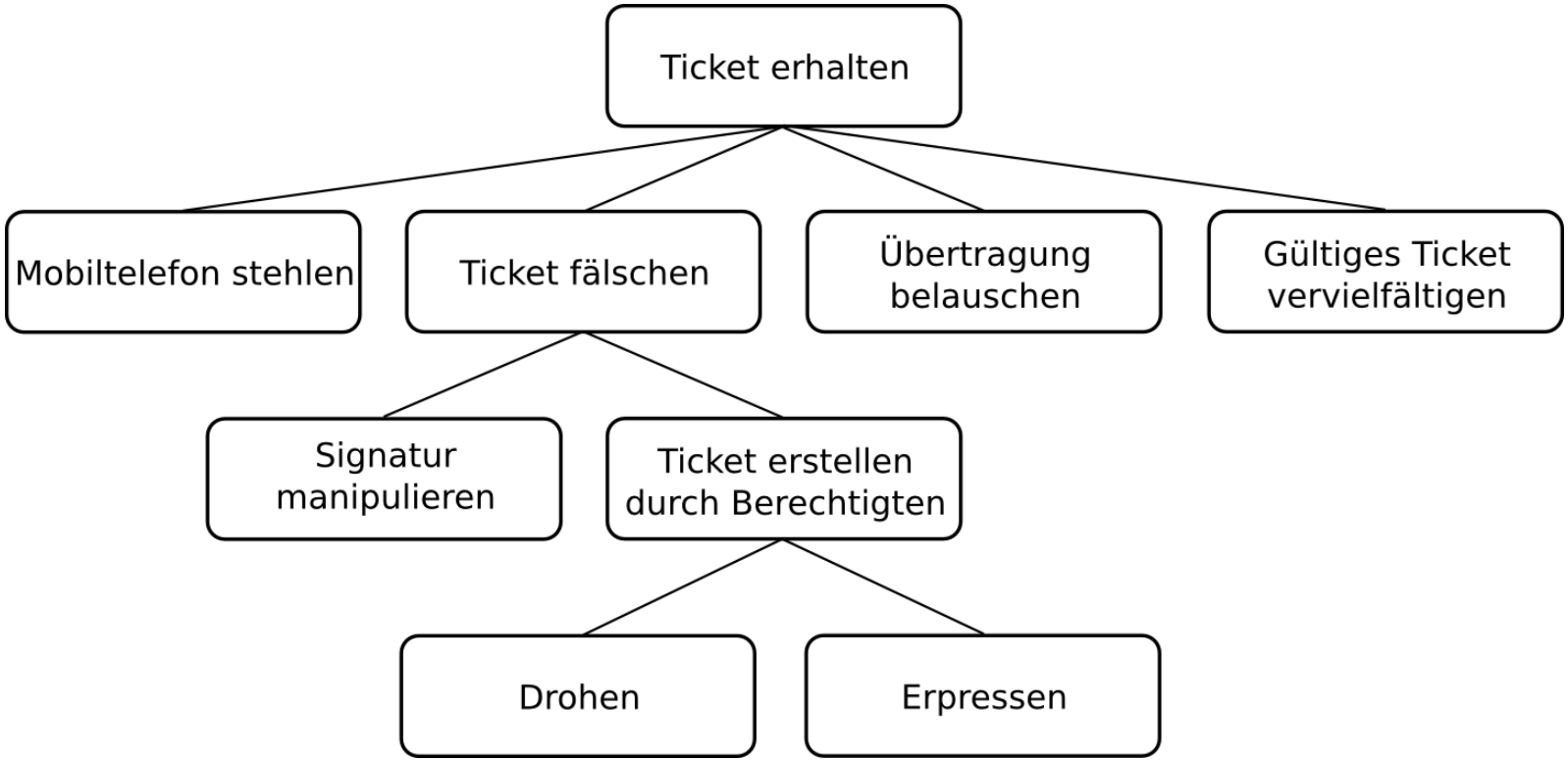
Bedrohungs- und Risikoanalyse

- Typisierung von Bedrohungen (z.B. nach)
  - Zielobjekt
  - UrheberIn
  - Motivation/Absicht
  - Wahrscheinlichkeit des Auftretens
  - Auswirkungen/Kosten
- Auflistung der Bedrohungen
- Risikoanalyse/Risikobewertung
- Maßnahmen
- Restrisikoabschätzung

Angriffsbäume/Attack Trees

- Technik zur Dokumentation von Bedrohungen
- Identifizierung möglicher Bedrohungen mit iterativer Detaillierung
- Vorgestellt von Bruce Schneier
- Darstellung erfolgt in Baumstruktur
- Knoten/Kanten können zur Bewertung der Eintrittswahrscheinlichkeit einer Bedrohung verwendet werden
- Erstellung ist manuell oder automatisch möglich
- Bei größeren Systemen werden Angriffsbäume schnell unübersichtlich. Z.B. Angriffsbaum für Netzwerk

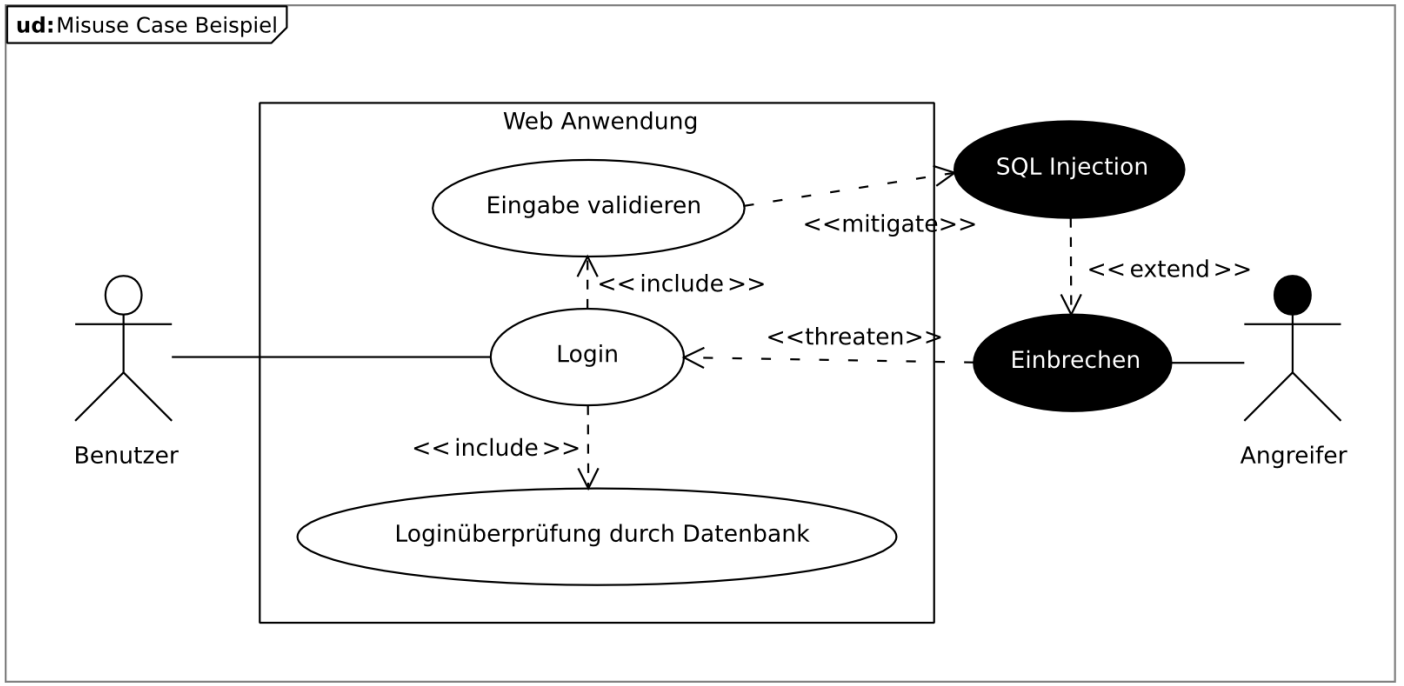
Beispiel für Angriffsbäume/Attack Trees



(Vergleiche Fankhauser, Schanes, Brem in Softwaretechnik, 2009)

Misuse Cases

- Notation von Sindre und Opdahl (2005) vorgestellt
- Akzeptanz von Use Cases vorhanden
- Misuse Cases noch nicht durchgesetzt; Keine Toolunterstützung



(Vergleiche Fankhauser, Schanes, Brem in Softwaretechnik, 2009)

Sicherheit in der Entwurfsphase

INSO – Industrial Software
Institut für Rechnergestützte Automation | Fakultät für Informatik | Technische Universität Wien

Architektur

(Vergleiche CERT, https://www.securecoding.cert.org)

Sicherheit in der Implementierung

INSO – Industrial Software
Institut für Rechnergestützte Automation | Fakultät für Informatik | Technische Universität Wien

Sichere Architektur vs. Sichere Programmierung

- Eine sichere Softwarearchitektur kann durch Programmierfehler unsicher werden
  - Fehlerhafte Prüfung von Zertifikaten
- Eine Software ohne Schwachstellen durch Programmierfehler kann Schwächen durch die Architektur aufweisen
  - Fehlerfreie telnet Implementierung ist durch unverschlüsselte Kommunikation angreifbar

Sichere Programmierung

- Auflistungen der häufigsten Sicherheitsfehler
  - OWASP: Top Ten Web Vulnerabilities
  - CWE und SANS: TOP 25 Most Dangerous Programming Errors
- Programmierrichtlinien
  - CERT Secure Coding Standards
  - Secure Coding Guidelines for the Java Programming Language
- Sicherheitskonzepte von Programmiersprachen
  - Automatische Längenprüfung und Speicherverwaltung, ...
  - Kenntnis über vorhandene Sicherheitskonzepte
  - Korrekte Anwendung der vorhandenen Sicherheitskonzepte

Auslieferung von Software

(Vergleiche `http://dilbert.com/strips/comic/2010-08-21/`)

Penetrate and Patch

Ansatz „Penetrate and Patch“ für sichere Software nicht ausreichend!

- Sicherheitsfehler werden nur im Betrieb gefunden und mit Patches behoben
- Wird von einigen Unternehmen noch immer betrieben
- Schwachstellen, welche von AngreiferInnen nicht gemeldet werden, können auch nicht behoben werden
- Patches beheben nicht immer die eigentliche Ursache, sondern oft nur Symptome
- Zeitfenster für Angriffe bleibt vorhanden, bis alle Systeme gepatcht werden. Bei einigen Systemen werden Patches nie eingespielt.
- Aus Patches können AngreiferInnen Details über die Schwachstelle ableiten

(Vergleiche John Viega und Gary McGraw, 2002)

SQL Injection – Fehlerhafter Code

Listing 1: action.php

```
$pw=$HTTP_GET_VARS["pw"];
...
$result=mysql_query("SELECT * FROM users
                     WHERE password='$pw'");
```

Listing 2: index.html

```
<form method="get" name="reg" action="action.php">
  Login: <input type="text" name="pw">
  <input type="submit" name="do" value="Login">
</form>
```

SQL Injection – Grundlagen

Eine Manipulation des SQL Statements durch einen Angreifer/eine AngreiferIn ist möglich.

- Einfaches Hochkomma: `te'st`. Fehlermeldung der Datenbank wird angezeigt. Schwachstelle erkennbar. Zusätzlich werden u.U. Informationen zur Datenbank und zum Aufbau des SQL Statements ausgegeben.
- Verwendung von Tautologien wie `' or '1'='1` was zu `SELECT * FROM users WHERE password='' or '1'='1'` führt
- Auslesen von Daten aus weiteren Tabellen mittels UNION oder Sub Queries
  - `' UNION SELECT * FROM system`

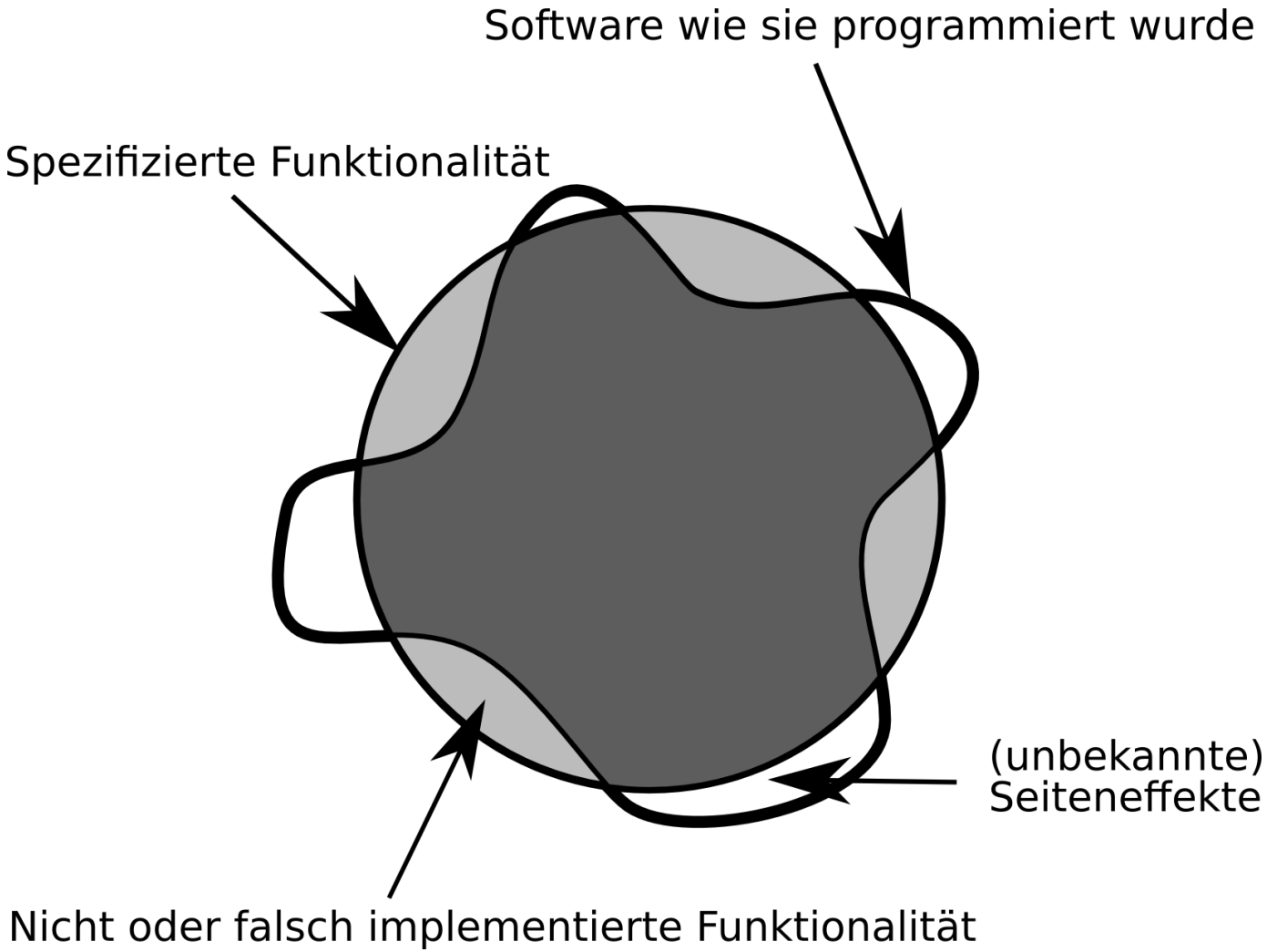
SQL Injection – Gegenmaßnahmen

- *Jede Eingabe* muss *validiert werden.* Es ist nicht ausreichend nur Formularfelder zu validieren. Injection ist auch zum Beispiel durch HTTP Header-Felder möglich.
- Filterung spezieller Datenbankzeichen wie z.B. einfache Hochkomma → Blacklist Filtering
- Besser: Prüfung der Eingaben auf gültige Zeichen → Whitelist Filtering
- Precompiled Statements verwenden. Das Statement wird in der Datenbank mit Platzhaltern kompiliert. Spätere Manipulation des Statements durch Eingaben nicht mehr möglich.
- Minimierung des Schadens bei Angriffen: Einschränkung der Rechte in der Datenbank → Principle of „Least Privilege“.

Sicherheitstests

INSO – Industrial Software
Institut für Rechnergestützte Automation | Fakultät für Informatik | Technische Universität Wien

Software Fehler



(Vergleiche Thompson, *Why security testing is hard*)

Security Features != Secure Features

- Sicherheitsrelevante Funktionalitäten
  - Anforderungen mit sicherheitskritischen Aspekten
  - Funktionale Fehler haben Auswirkung auf Sicherheit
  - Durchführen von funktionalen Tests
  - Beispiel: Authentifizierung, Public Key Infrastrukturen, ...
- Sichere Funktionen
  - Sicherheitsfehler können in jeder Funktionalität auftreten
  - Testen der Robustheit gegen Angriffe
  - Beispiel: Fehlende Input Validierung
- Wird oft verwechselt $\rightarrow$ klare Definition und Abgrenzung erforderlich

Sicherheitstests

- Studie Carnegie-Mellon-Universität: „tausend Zeilen Programm ca. fünf bis fünfzehn Bugs – nach dem Testen“
  - Diese bleiben oft unentdeckt
  - Haben keine Auswirkung auf die restliche Funktionalität
  - Stellen mögliche Sicherheitsfehler dar
  - Gezielte Suche nach diesen Fehlern erforderlich
- Software hat heute oft viele Millionen Lines of Code
- Test ist eine Momentaufnahme und zeitlich beschränkt → AngreiferIn hat beliebig Zeit
- Testen alleine nicht ausreichend für ein sicheres System
- Testen ist ein Teil zur Erstellung eines sicheren Systems

Projektmanagement in Bezug auf Software-Qualität

(Vergleiche `http://dilbert.com/strips/comic/1995-11-13/`)

Sicherheit während des Betriebs

INSO – Industrial Software
Institut für Rechnergestützte Automation | Fakultät für Informatik | Technische Universität Wien

Betrieb

- Es zeigt sich, ob Sicherheitsanforderungen tatsächlich erfüllt werden

- Kenntnis von Sicherheitsanforderungen für den Betrieb
- Definition und Einhaltung des Service Levels
- Bemerken von Sicherheitsvorfällen
- Mögliche Reaktionen auf Sicherheitsvorfälle
- Information Technology Infrastructure Library (ITIL)
- Organisatorische Sicherheitsanforderungen

- Sichere, kontrollierte Außerbetriebnahme!

Zusammenfassung

- Sicherheitsprobleme treten auf!
- Sicherheit in allen Phasen des Softwareentwicklungsprozesses
- Sicherheitsziele Vertraulichkeit, Integrität, Verfügbarkeit
- Sicherheitsanforderungen und geeignete Sicherheitsmaßnahmen
- Sicherheit ist ein Prozess

- Minimaler Einblick in IT-Security
- Weitere Lehrveranstaltungen der ESSE besuchen :-)

Mit auf den Weg...

- Security ist Management-Thema, muss im Projekt entsprechend geplant werden!
- Security von Beginn an berücksichtigen, bereits beim Entwurf, über alle Phasen einer Software hindurch, bis zum Betrieb und Ausserbetriebnahme
- Design Principles (siehe Saltzer/Schroeder 1975)
- Verwendung von Standards
- Lernen von bekannten Sicherheitslücken
- Testen auf Sicherheit

- Für die Programmierung: *Input Validierung!*

Literatur – 1/3

- Florian Fankhauser, Christian Schanes, und Christian Brem. Sicherheit in der Softwareentwicklung. In *Softwaretechnik - Mit Fallbeispielen aus realen Entwicklungsprojekten*, Kapitel 13, Seiten 589–646. Pearson Studium, München, 1. Auflage, 2009. `http://www.inso.tuwien.ac.at/publications/softwaretechnik/`
- Ross Anderson. *Security Engineering. A Guide to Building Dependable Distributed Systems*. Wiley Publishing, Inc., 2. Auflage, 2008. ISBN 978-0-470-06852-6. `http://www.cl.cam.ac.uk/~rja14/book.html`

Literatur – 2/3

- Ed Skoudis und Tom Liston. *Counter Hack Reloaded. A Step-by-Step Guide to Computer Attacks and Effective Defenses*. Pearson Education, Inc., 2. Auflage, 2006. ISBN 0-13-148104-5
- Bruce Schneier. *Secrets & Lies: Digital Security in a Networked World.* Wiley Publishing, Inc., Indianapolis, Indiana, 2004. ISBN 0-471-45380-3
- Bruce Schneier. *Schneier on Security*. Wiley Publishing, Inc., Indianapolis, Indiana, 2008. ISBN 978-0-470-39535-6
- John Viega und Gary McGraw. *Building Secure Software: How to Avoid Security Problems the Right Way*. Addison-Wesley, 2002

Literatur – 3/3

- Stuart R. Faulk. Software Requirements: A Tutorial. Technischer Bericht, Center for High Assurance Computer Systems, US Navy, 1995. `http://www.cs.umd.edu/class/spring2004/cmsc838p/Requirements/Faulk_Req_Tut.pdf`
- Jerome H. Saltzer und Michael D. Schroeder. The protection of information in computer systems. In *Proceedings of the IEEE*, Band 63, Seiten 1278–1308, 1975

Vielen Dank!

esse@inso.tuwien.ac.at

http://security.inso.tuwien.ac.at/

INSO – Industrial Software
Institut für Rechnergestützte Automation | Fakultät für Informatik | Technische Universität Wien