
FernUniversität
in Hagen

Fakultät für Mathematik und Informatik

# Hausarbeit

**im Seminar 21817 „IT-Sicherheit“**

| | |
|---|---|
| **Thema:** | Aspekte der Sicherheit in der Programmierung |
| **Autor:** | Florian Mahlecke <florian.mahlecke@cirosec.de><br>MatNr. 8632014 |
| **Autor:** | Kirsten Katharina Roschanski <studium@kirsten-roschanski.de><br>MatNr. 9053522 |
| **Version vom:** | 22. Mai 2013 |
| **Betreuer:** | Dipl. Inf. Daniel Berg |

## Eidesstattliche Erklärung

# Eidesstattliche Erklärung zur Hausarbeit

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

$Unterschrift$ : $Ort, Datum$ :

## Eidesstattliche Erklärung

# Eidesstattliche Erklärung zur Hausarbeit

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

$Unterschrift$ : $Ort, Datum$ :

# Inhaltsverzeichnis

# 1 Einleitung

In kommerziellen Softwareprojekten arbeitet ein geschlossener Kreis an Personen an dem Code. Dieser wird dann in der Regel vor Veröffentlichung durch entsprechende Experten geprüft und freigegeben.
Bei OpenSource Projekten, wo der der Quellcode frei zugänglich ist und eine Mitarbeit auch gewünscht ist, kann hingegen jeder der programmieren kann oder auch denkt den Code erweitern und eigene Module entwickeln.
Bei vielen Modulen wird dabei oft die API ignoriert oder es schleichen sich schnell kritische Sicherheitlücken ein.
In der nachfolgenden Hausarbeit geht es um die Sicherheit in der Programmierung. Dabei soll auf typische Fehler eingegangen werden und anhand von unterschiedlichen Angriffsszenarien, Techniken und Technologien gezeigt werden wo die häufigsten Schwachstellen liegen und wie man diese ganz einfach beseitigen kann. Ausgangspunkt ist die Modulare Objekt orientierter Programmierung, die heute in fast allen Softwareprojekten zur Anwendung kommt.

# 2 Technologieübergreifende Schwachstellen

- Race Conditions
- Sichere Kommunikationswege
- Kryptografie

## 2.1 Unterscheidung in Technologien

- Webanwendungen
- "FatClient"-Anwendungen

## 2.2 Typische webbasierte Schwachstellen

- Cross-Site Scripting
  Erläuterung von XSS
  **Gegenmaßnahme:** Ein- und Ausgabevalidierung
- Cross-Site-Request Forgery
  Erläuterung von CSRF
  **Gegenmaßnahme:** Serverseitig ausgestelltes CSRF Token

- SQL Injection
  Erläuterung von SQL-Injection
  **Gegenmaßnahme:** Eingabevalidierung

- Clientseitige Sicherheitsmechanismen
  Erläuterung von clientseitig implementierten Sicherheitsmechanismen z.B. im Kontext von JS **Gegenmaßnahme:** Serverseitige Validierung von Eingabeparametern

- Session-Management Erläuterung von Schwachstellen im Session Management einer Webapplikation
  **Gegenmaßnahme:** Cookie Flags verwenden, sinnvolle Session Timeouts, etc.

- Optional: Technische Gegenmaßnahmen, Einführung einer Webapplication-Firewall oder einen OS Produkt wie "mod_security"

- Optional: Erläuterung von "Security"-Frameworks im Webumfeld wie z.B. Apache Wicket

- Optional: Was bringt HTML5 für neue, möglicherweise sicherheitskritische Features mit sich?
  - Cross Origin Ressource Sharing: Erleichtert XSS ("Shell of the Future", "Beef"-Framework)
  - Support von SVG-Grafiken: Einbetten von JS Code in Grafiken die innerhalb der Applikation nachgeladen werden
  - Webstorage: Speicherung von sensitiven Daten innerhalb des Webstorage des Browsers, Zugriff per XSS auf den Webstorage

## 2.3 Typische "FatClient"-Schwachstellen

- BufferOverflow
  Erläuterung von BufferOverflows
  **Gegenmaßnahme:** Verwendung von sicheren Funktionen

- Als weitere Gegenmaßnahme für Buffer Overflows die Funktion von ASLR erläutern

- Verwendung von proprietärer Kryptografie
  Erläuterung von Schwierigkeiten durch den Verwendung von proprietärer Kryptografie
  **Gegenmaßnahme:** Auf offene Algorithmen (Standards) zurückgreifen

- Optional: Querverweis auf Microsofts SDL

- Optional einen Querverweis auf Tools die im Nachgang potentiell unsichere Anwendungen absichern z.B. MS EMET

### 2.4 Technologieübergreifende Lösungsansätze

- Regelmäßige Durchführung von Penetrationstest
- Einführung von Standards, im Kontext von Anwendungssicherheit vielleicht einen Querverweis auf ISO 27001. Natürliche keine vollständige Einführung eines ISMS sonder eher den PDCA-Zyklus betrachten
- Einführung von Bugtracking-Systemen
- Durchführung von Awareness-Maßnahmen
- Einführung von Entwicklungsrichtlinien

## 3 Programmierfehler

### Überblick

In diesem Kapietel soll es um oft gemacht und gern immer wieder auftretenden Programmierfehler gehen. Dabei soll der Schwerpunkt auf cross site scripting (XSS) gelegt werden und die Verwendung von Access modifiers in der Objektorientierten Programmierung.

Public - If you can see the class, then you can see the method
Private - If you are part of the class, then you can see the method, otherwise not.
Protected - Same as Private, plus all descendants can also see the method.
Static (class) - Remember the distinction between "Class" and "Object" ? Forget all that. They are the same with "static"... the class is the one-and-only instance of itself.
Static (method) - Whenever you use this method, it will have a frame of reference independent of the actual instance of the class it is part of.

## 4 Angriffssenarien für bekannte Sicerheitslücken

### Überblick

In diesem Kapietel soll es um bekannte Sicherheitslücken gehen, die häufig von Hackern als erstes Angriffsziehl dienen.

# 5 Sicherer Code und Robuste Programmierung

## Überblick

Hier soll der Frage nachgegengen werden, welche Bedrohungen von schadhaften Code ausgehen.

# 6 Fazit

# Zusammenfassung

# Abstract

# Anhang

# Listingverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis