University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Apsi Lab 2

18. Dezember 2013

put abstract here

Inhaltsverzeichnis

1 Robust Programming **2**
1.1 Model . . . 2
1.1.1 XXS, SQL-Injection und Validierung . . . 2
1.1.2 Schutz der Model-Klasse vor schädlichem Missbrauch . . . 2
1.1.3 Postleitzahl Validierung . . . 2
1.2 Controller . . . 2
1.2.1 SSL . . . 2

2 Weitere Mögliche Massnahmen **2**
2.1 DOS-Attacke . . . 2

Institute of computer science,
www.fhnw.ch

1 Robust Programming

Folgende

1.1 Model

1.1.1 XXS, SQL-Injection und Validierung

1.1.2 Schutz der Model-Klasse vor schädlichem Missbrauch

Die schwierigkeit einer korrekten Login-implementation liegt darin, den Ablauf so zu gestalten dass nur dieser Ablauf zu einer validen Änderung führt. Heisst nur beim vorgesehenen Ablauf können nur korrekte Daten geschrieben werden, alles andere darf die Datenbank nicht verändern. So sind beispielsweise setters auf das Passwort problematisch. Zusammen mit der $.save()$ Methode könnte ein Fehlerhafter oder bösartiger Code das Passwort eines beliebigen Users überschreiben. Usernamen und ID sind ebenfalls Felder, auf die nur lesend zugegriffen werden soll.

Umsetzung: Um das Model korrekt umzusetzen wurden zwei Konstruktoren implementiert. Ein Konstruktor nimmt einen Benutzernamen und Passwort entgegen. Dieser ist zuständig, einen neue erstellten Benutzer zu repräsentieren. Ein zweiter Konstruktor nimmt eine ID und ein Passwort entgegen.

1.1.3 Postleitzahl Validierung

Die Postleitzahl wird wie verlangt, durch einen externen Service überprüft. Die Website fragt post.ch ab,ob eine gewisse Postleitzahl existiert. Wenn diese nicht existiert, enthält die HTML Antwort den String K̈eine PLZ Gefundenünd wir lassen diese PLZ nicht zu. Wir überprüfen nur, ob die Postleitzahl existiert und nicht ob die angegebene Stadt mit der Postleitzahl übereinstimmt.

1.2 Controller

Der Controller wurde so simpel wie möglich gehalten,

1.2.1 SSL

2 Weitere Mögliche Massnahmen

2.1 DOS-Attacke