




Post It

Modul 151

Zug GIBZ

23.05.2022

Robin Ruzza & Julia Sorrentino



Inhaltsverzeichnis

Informieren:	4
Aufgabenstellung:	4
Planen:	4
Zieldefinition:	4
Meilensteine:	5
Entscheiden:	5
Realisieren:	5
4-Tier Architektur	5
Datenbankanbindung/CRUD	6
Datensicherheit:	6
Angriffsmöglichkeiten	6
-Sicherheitslücken(im Web-Server, Betriebssystem)	6
-HTML Injection	6
-SQL Injection	6
-Cross Site Scripting	6
-Cross Site Request Forgery (CSRF)	6
-Session Hijacking	6
Der Angreifer muss nun eine Möglichkeit finden, die URL mit der SessionID des Benutzers auf seinem PC darzustellen. Dies kann durch mitschneiden des Netzwerkverkehrs erfolgen oder durch das Einschleusen eines einfachen JavaScript-Befehls beziehungsweise HTMLTags, der in einer Webseite auf dem Webserver platziert wird.	6
-Social Engineering	6
-Directory Traversal	6
-Denial of Service, Distributed Denial of Service (DoS)	6
-E-Mail Injection	6
-Man in the Middle Attack	7
-Phishing	7
-Shell Injection	7
-Brute-Force	7
- usw.	7
Ergriffene Massnahmen	7
1. Passwort Mindestanforderungen:	7
2. Encryption vom Passwort	7
3. E-Mail-Validierung	7
4. Https Verschlüsselung / SSL	7
Journal :	7

Tag 1:	7
Tag 2:	8
Tag 3:	8
Tag 4:	8
Tag 5:	9
Kontrollieren:	9
.....	10
Auswerten:	10

Informieren:

Aufgabenstellung:

Für Ihre Applikation erstellen Sie ein GIT Repository. In der Dokumentation halten Sie fest welche Features/Funktionen Teile der Software von wem realisiert werden.

Diese Liste sollte mit Ihren regelmässigen Commits im Software Repo übereinstimmen.

Es wird erwartet das jeder Schüler/jede Schülerin mindestens 5 Gehaltvolle Commits leistet. Ein Commit zählt nur wenn er wesentliche Änderungen enthält.

Gibt es an einem Schultag keine Code-Änderungen dokumentieren Sie das in Ihrer Dokumentation. (Journal Ihrer Arbeiten)

Die Projektarbeit realisieren Sie in einer ausgewählten Technologie. Sie verwenden generelle Techniken wie HTML, CSS und JavaScript. Wenn Sie Frameworks einsetzen, begründen Sie die Wahl in der Dokumentation.

Für die Bewertung der Arbeit werden unter anderem folgende Kriterien beachtet:	Titel, Name, Thema, Datum, Seitenzahl usw.
Formale Anforderungen	
Generelle Informationen	Spezifikation / Zieldefinition / Meilensteine
Architektur	Min. 4-Tier Architektur Datenbankanbindung / CRUD Datensicherheit wird berücksichtigt (Themen aus «Sicherheit in Web-Applikationen») Sessions / Transationen / SSL/TLS
Arbeitsjournal	Wer hat wann was gemacht? Es muss ersichtlich sein wer welche Arbeiten gemacht hat.
Datensicherheit	Ihre Anwendung ist gegen die gelernten Angriffe geschützt. In der Dokumentation gehen Sie auf die verschiedenen Angriffsmöglichkeiten sowie die ergriffenen Massnahmen ein.

Sie dokumentieren alle Ihre Entscheide und Arbeiten in einem Dokument. Die Bewertung Ihrer Leistung findet Anhand des Dokuments statt.

Sie präsentieren Ihre Arbeit Ihren Kollegen. Bewertet werden folgende Kriterien

- Die Präsentation startet nach maximal zwei Minuten Vorbereitungszeit
- Die Präsentation dauert maximal 5 Minuten
- Einsatz von Präsentationsmitteln (z.B. Beamer, Tafel, Whiteboard usw.)
- Allgemeine Präsentationsmerkmale (z.B. Blick in Klasse, deutliches Sprechen usw.)

Planen:

Zieldefinition:

Unser Hauptziel ist es eine Vollständige und professionelle Dokumentation für unsere Software herzustellen. Dabei ist es wichtig, dass wir unser Journal bei jeglichen Änderungen/Erweiterungen/Updates von unserer Software festhalten und Dokumentieren. Wir müssen dabei das Abgabedatum sehr im Auge behalten, da es durch eine verspätete Abgabe einen Noten Abzug gibt.

Wir müssen auch jede Woche mindestens ein Commit machen, und darauf achten, dass jeder Schüler mindestens 5 Commits macht, somit müsste jeder einmal pro Woche ein Commit machen, damit alles aufgeht.

Ein weiterer wichtiger Teil dieses Projektes ist die Datensicherheit. Wir müssen unsere Software so gestalten, dass Angriffe nicht bis nur leicht möglich sind. Die Massnahmen, die wir einführen werden, werden von Julia recherchiert und dokumentiert, und von Robin umgesetzt.

Meilensteine:

1. Das Planen und Entscheiden von unserem Vorgehen
2. Das Grundgerüst der Software erstellen
3. Die Datenbank mit der Webseite verbinden
4. Die Datensicherheit zu recherchieren und in unserer Software anzuwenden
5. Das Testen unsere Software und das Abschliessen des Projektes
6. Die Dokumentation mit den letzten Informationen füllen

Entscheiden:

Wir haben uns für die IPERKA Methode entschieden und diese Methode auch angepasst, damit wir ein Journal führen können und die Theorie, die wir benutzt haben, erklären und begründen können.

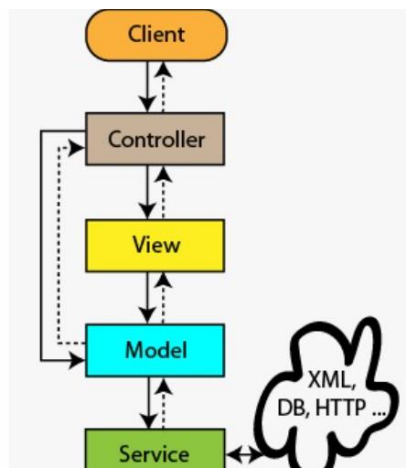
Wir haben uns für die IPERKA Methode entschieden, da wir beide uns am meisten mit diesem Vorgehen auskennen und da es sehr übersichtlich ist. Bei der Datensicherheit werden wir uns eher auf die Sicherheit der Passwörter neigen. Wir haben uns dafür entschieden unsere Webseite mit Passwort Verschlüsselung, und Passwort minimal Anforderungen zu Schützen. Wir wollen ebenfalls eine E-Mail-Verifizierung machen, die verifiziert, ob die angegebenen E-Mail Adresse echt ist.

Um die Webseite und deren Daten zu Schützen wollen wir eine Https Verschlüsselung machen. Die Dokumentation und Theorie zu diesen Angriffsmöglichkeiten und unsere Ergriffenen Massnahmen findet man im «Datensicherheit» Abschnitt unserer Dokumentation.

Realisieren:

4-Tier Architektur

Bei der 4-Tier Architektur (auch 4-Schichten-Architektur) geht es darum die Prinzipien zur Strukturierung von Software-Architekturen zu definieren. Dabei ist jede Schicht einem bestimmten Aspekt zugeordnet z.b. eine Teil-Funktionalität, eine geschlossene Komponente oder eine Klasse. Das Haupt Ziel der Schichtenmodelle/ Schichtenarchitektur ist es, die Komplexität der Software-Systemen besser zu strukturieren und reduzieren zu können.



Datenbankanbindung/CRUD

Die Datenbank haben wir über das Microsoft Entity Framework direkt im Code gemacht. Somit mussten wir auch kein SQL schreiben.



Das Entity Framework (auch EF genannt), ist ein Framework für objektrelationale Abbildung. Wie man auf dem Bild sieht, wurde es von Microsoft entwickelt und dient dem ORM auf .Net-Framework- sowie auf .NET-Objektstrukturen

Wir haben das Framework verwendet da es eine einfache Verwaltung der Datenbank direkt aus dem Code ermöglicht und wir direkt aus Klassen Tabellen in der DB erstellen können

Datensicherheit:

Angriffsmöglichkeiten

-HTML Injection

Dur HTML in Benutzereingaben das HTML der Webseite manipulieren.

-SQL Injection

Durch Metazeichen in Benutzereingaben SQL-Aufrufe manipulieren.

-Cross Site Scripting

Einschleusen von manipulierten Informationen.

-Cross Site Request Forgery (CSRF)

Eine Cross-Site-Request-Forgery (XSRF/CSRF) ist ein Angriff auf ein Computersystem, bei dem der Angreifer unberechtigt Daten in einer Webanwendung verändert. Er bedient sich dazu eines Opfers, das ein berechtigter Benutzer der Webanwendung sein muss. Dabei muss der Angreifer das Opfer dazu bringen, manipulierte HTTP-Requests auszuführen.

-Session Hijacking

Der Angreifer muss nun eine Möglichkeit finden, die URL mit der SessionID des Benutzers auf seinem PC darzustellen. Dies kann durch mitschneiden des Netzwerkverkehrs erfolgen oder durch das Einschleusen eines einfachen JavaScript-Befehls beziehungsweise HTMLTags, der in einer Webseite auf dem Webserver platziert wird.

-Directory Traversal

Manipulierte Pfadangaben. Und somit Files mit Daten stehlen.

-Denial of Service, Distributed Denial of Service (DoS)

Vielzahl von Verbindungsanfragen die versuchen die Ressourcen für die regulären Anfragen einzuschränken / zu entziehen

-E-Mail Injection

Angreifer fügt manipulierte Daten in Kontaktformular ein. Wird meist für den Versand von SPAM missbraucht

-Man in the Middle Attack

Angreifer versucht sich zwischen Webserver und Benutzer einzuklinken und dabei mit dem Benutzer eine Verbindung aufzubauen ohne dass dieser das merkt. Zuerst muss einem Benutzer eine gefälschte URL untergejubelt werden, z.B. via Mail oder Link in einem Forum/Blog.

-Phishing

Meist massenweise per E-Mail gesendete Aufforderungen Zugangsdaten für z.B.: Online-Banking etc. auf einer Webseite einzugeben. Die Seite sieht äusserlich wie die Webseite des Betreibers aus, unterliegt aber der Kontrolle des Angreifers.

-Shell Injection

Ausführen von Konsolenbefehlen auf dem Server über Missbrauch eines ungesicherten oder selber eingeschleusten "shell()"-Befehls (PHP).

-Brute-Force

Einfache Passwörter können durch Brute-Force-Attacken, also durch zufälliges Ausprobieren (auch durch automatisches Prübeln durch Roboter mit entsprechenden Wörterbüchern von häufig verwendeten Passwörtern) geknackt werden.

- usw.

Ergriffene Massnahmen

1. Passwort Mindestanforderungen:

Um zu verhindern das eine Person das Passwort unserem User «errät» haben wir eine Mindestanforderung für das Passwort gemacht. Dabei muss das Passwort mindestens 8 Zeichen lang sein, das Passwort sollte mindestens ein kleiner Buchstabe und ein grosser Buchstabe haben. Ebenfalls ist mindestens ein Sonderzeichen eine Anforderung.

2. Encryptung vom Passwort

Passwort wird nicht im Klartext abgelegt sondern wird mithilfe eines Hashes verschlüsselt bevor es in die Datenbank gespeichert wird

3. E-Mail-Validierung

Da es eventuell Benutzer gibt, die eine falsche oder nichtexistierende E-Mail-Adresse angeben haben wir eine Basic E-Mail Validierung hinzugefügt. Dabei wird die E-Mail-Adresse erst akzeptiert, wenn es dieses Format besitzt: [variable@variable.TLD](#) (Das TLD steht für Top Level Domain)

4. Https Verschlüsselung / SSL

Unsere Webseite ist durch HTTPS mit SSL verschlüsselt. Und somit gegen einige der oben genannten Attacken geschützt.

Journal :

Tag 1:

Als erstes haben wir uns zusammengesessen und entschieden welches Vorgehen wir benutzen wollten und wie haben uns für IPERKA entschieden. Danach haben wir die ersten kleineren Aufgaben aufgeteilt. Julia würde Grossteils die Dokumentation und die Theorie übernehmen während Robin sich um die Software kümmert. Als erstes fing Julia mit dem Grundgerüst der Dokumentation (IPERKA) an, und füllte dies mit den Anfangsinformationen (Das Informieren, das Planen und die Zieldefinition und der erste Tag des Journals)

Tag 2:

Heute hat Julia mit der Dokumentation angefangen und hat sich mit dem zweiten Teil der Planung befasst, und hat somit die Meilensteine erstellt und diese mit Robin besprochen, und angepasst. Somit haben wir nun 6 Meilensteine. Nachdem der Planungsteil beendet war, und wir unser Ziel und Meilensteine wussten, hat Julia mit dem Testprotokoll angefangen, damit wir am Ende des Projektes die Ziele und Meilensteine testen können. Wir haben besprochen, ob wir nur ein Testprotokoll für die Software machen wollen, oder ob wir zwei Testprotokolle machen wollen. Eines für die Software inkl. der Datensicherheit und Datenbank und ein Testprotokoll für die allgemeine Organisation, Administration und Dokumentation.

Wir haben uns schlussendlich für ein einziges Testprotokoll entschieden, darin sollte die Software, die Datenbank und die Datensicherheit getestet werden.

Und somit, hat Julia nach der Besprechung mit dem Testprotokoll angefangen und hat zuerst das Grundgerüst des Testprotokolls gemacht. Dies hat sie danach auch zusammen mit der Dokumentation auf Github hochgeladen. Ihr ist dabei aufgefallen, dass sie für das Committen von Files noch Hilfe brauchte, und somit schaute Sie auf [YouTube](#) ein kurzer Tutorial an.

Robin hat in der Zwischenzeit das Projekt erstellt. Danach hat er das erste Identity model erstellt. Als nächstes fing er mit den views für die Login und der Registration, dabei hat er natürlicherweise mit Visual Studio gearbeitet und mit c# und Razor programmiert.

Der Code wurde kommentiert, damit es einfacher nachvollziehbar ist.

Tag 3:

Da hatten wir beide Frei und haben somit nicht an diesem Project gearbeitet.

Tag 4:

Heute hat Julia als erstes mit dem Testprotokoll angefangen und hat diesen ausgefüllt, am Ende besprach sie noch ob Testfälle fehlten oder ob man etwas bestimmtes anpassen musste. Die meisten Ziele/Funktionen, die von uns getestet werden, sind die Allgemeinen Anforderungen und ein paar Sicherheitsmassnahmen, die wir zusammen entschieden haben. Das Testprotokoll befindet sich im «Kontrollieren» Abschnitt unserer Dokumentation. Das File der Testfälle findet man auch in unserem GitHub unter dem «Dokumentation» Folder.

Beim Lauschen eines Gespräches von anderen Klassenkameraden ist bei uns der Zweifel der Commits aufgetaucht, diese meinten, dass die Commits nur Codebasiert sind, und dass somit jede Person, je 5 Commits im Code machen musste. Um dies zu klären haben wir nachgefragt und es stellte sich heraus, dass z.b. Änderungen und der Dokumentation oder Erstellen eines Testings Dokumentes nicht zählen würde. Diese Information stellte unsere Planung sehr auf den Kopf, da wir geplant hatten, dass sich Julia mit der Theorie, der Dokumentation und den Testfällen befassen würde und sich Robin um die Software kümmert.

Da die Dokumentation somit nicht zu den Commits zählten haben wir diese zusammen mit den Testfällen aus dem GitHub entfernt und dies werden wir am Ende als Endprodukt hochladen, da das ständige Aktualisieren und Pushen der Dokumente auch Zeit kosten.

Damit Julia auch mindestens 5 Commits hat, hat Robin Ihr einen Auftrag gegeben. Dabei musste sie die Post Klasse erstellen damit das Posten funktionieren würde. Um dies zu tun, brauchte Julia erstmals noch den .NET Framework 4.8. Dies musste sie installieren und updaten was lange lief. Doch schlussendlich funktionierte es und sie konnte mit dem Programmieren anfangen. Und pushte den Code auf GitHub.

In der Zwischenzeit hat Robin angefangen die Differenzierung der Projekt Architektur aufzubauen. Als er fertig war, hat er noch mit dem datahandling begonnen und das Model für den Post erstellt. Der Code sollte selbstverständlich sein, wenn man die Kommentare beachtet.

Tag 5:

Heute hat Julia noch die letzten Teile der Dokumentation gemacht, dazu gehören der Teil der Datensicherheit, die Datenbankbindung, das Kontrollieren/Testen der Webseite und das Auswerten des gesamte Projekt. Bei der Datensicherheit war uns unklar ob wir alle möglichen (uns bekannten) Angriffsmöglichkeiten recherchieren mussten, und so fragten wir nach. Es stellte sich heraus, dass wir doch alle möglichen Angriffsmöglichkeiten Beschreiben/erklären müssen. Beim Testen lief alles gut und wir konnten alle Testfälle durchführen. Das ausgefüllte Testprotokoll befindet sich im «Kontrollieren» Abschnitt unserer Dokumentation. Robin hat heute mit Julia noch die Webseite finalisiert und alles ausführlich kommentiert und letzte einträge in die Dokumentation getätigt. Am Schluss wurde die Dokumentation und das Test File mit dem Code auf GitHub hochgeladen.

Kontrollieren:

Dies ist unser noch nicht ausgefüllte Protokoll.

Wie man sieht, sind die meisten Testfälle über die Login und Registationsseite. Als Datensicherheit werden wir die Passwörter und die E-Mail-Verifizierung testen.

Julia Sorrent Version 0.1		13.06.2022	
Testfall Nr.	Erwartetes Verhalten	Eigentliches Verhalten	Erfüllt ?
1	Der Code enthält HTML		
2	Der Code enthält CSS		
3	Der Code enthält Javascript		
4	Es hat eine Funktionierende Datenbank		
5	Es hat eine Registrierungsseite die einen Account erstellt.		
6	Es hat eine Login Seite.		
7	Bei der Registration werden E-Mail adresse und Passwort erfasst.		
8	Dabei wird die E-Mail adresse validiert.		
9	Beim Login kann man sich mit den korrekten Daten (Passwort und E-Mail adresse) anmelden.		
10	Man kann einen Text posten		
11	Der Text wird von andere Benutzer auch gesehen.		
12	Es gibt pro Post ein Zeichenlimit von 300 Zeichen.		
13	Man kann nur Text posten und keine Bilder, videos, usw.		
14	Der Account kann gelöscht werden		
15	Das Passwort kann man abändern		
16	Das Passwort wird beim eingeben ge-censored damit es sicherer ist		
17	Das Passwort hat bestimmte anforderungen (muss min. 6 Zeichen lang sein)		
18	Das Passwort hat bestimmte anforderungen (muss min. 1 Zahl, Grossbuchstabe, Kleinbuchstabe & Sonderzeichen haben)		
19			

Auf der nächsten Seite ist unser ausgefülltes Testing Dokument, Robin hat die Tests gemacht und das Dokument ausgefüllt. Wie man sieht, haben alle Testfälle genau so funktioniert wie wir uns das vorgestellt haben. Und somit wäre das Testing beendet.

Testfall Nr.	Erwartetes Verhalten	Eigentliches Verhalten	Erfüllt ?
1	Der Code enthält HTML	Der Code enthält HTML	Ja
2	Der Code enthält CSS	Der Code enthält CSS	Ja
3	Der Code enthält Javascript	Der Code enthält Javascript	Ja
4	Es hat eine Funktionierende Datenbank	Es hat eine Funktionierende	Ja
5	Es hat eine Registrationsseite die einen Account erstellt	Es hat eine Registrationsseite die einen Account erstellt	Ja
6	Es hat eine Login Seite	Es hat eine Login Seite	Ja
7	Bei der Registration werden E-Mail adresse und Passwort erfasst	Bei der Registration werden E-Mail adresse und Passwort erfasst	Ja
8	Dabei wird die E-Mail adresse validiert	Dabei wird die E-Mail adresse validiert	Ja
9	Das Passwort kann man abändern	Das Passwort kann man abändern	Ja
10	Das Passwort wird beim eingeben ge-censored damit es sicherer ist	Das Passwort wird beim eingeben ge-censored damit es sicherer ist	Ja
11	Das Passwort hat bestimmte anforderungen (muss 8 Zeichen lang sein)	Das Passwort hat bestimmte anforderungen (muss 8 Zeichen lang sein)	Ja
12	Beim Login kann man sich mit den korrekten Daten (Passwort und E-Mail adresse) anmelden.	Beim Login kann man sich mit den korrekten Daten (Passwort und E-Mail adresse) anmelden.	Ja
13	Man kann einen Text posten	Man kann einen Text posten	Ja
14	Der Text wird von andere Benutzer auch gesehen	Der Text wird von andere Benutzer auch gesehen	Ja
15	Es gibt pro Post ein Zeichenlimit von 300 Zeichen	Es gibt pro Post ein Zeichenlimit von 300 Zeichen	Ja
16	Man kann nur Text posten und keine Bilder, videos, usw.	Man kann nur Text posten und keine Bilder, videos, usw.	Ja
17	Der Account kann gelöscht werden	Der Account kann gelöscht werden	Ja

Auswerten:

Rückblickend hat das meiste recht gut funktioniert, die Dokumentation und die Software haben wir fertiggekiegt. Die Zeit für den Auftrag war unserer Meinung nach recht Knapp, da man eine vollständige Planung, Dokumentation mit Journal, Testfälle, eine Datenbank, das Coden in wenigen Wochen machen musste.

Wir hatten jedoch ein paar Schwierigkeiten, diese basierten auf Missverständnisse. Eines der grössten Missverständnisse, war der Commit Teil, dieses Missverständnis setzte unser Projekt recht auf den Kopf. Da wir geplant hatten die Theorie, Dokumente, Administratorisches und das Coden aufzuteilen und dies auch so bis zur vorletzten Woche gemacht hatten, war es sehr schwierig am Ende noch einen Weg zu finden für Julia 5 sinnvolle Commits im Code zu machen.

Die Datensicherheit in unserer Webseite einzufügen war auch recht schwer, da wir uns in der vorherigen Aufgabe mit dem Social Engineering auseinandergesetzt haben und dieses Wissen nicht wirklich für unsere Webseite benutzen konnten. Jedoch achteten wir darauf die Basic Datensicherheitsmassnahmen für E-Mail und Passwort einzuhalten.