HOCHSCHULE
HANNOVER
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS
–
Fakultät IV
Wirtschaft und
Informatik

Bericht: Maßnahmen zur Qualitätssicherung

Dennis Grabowski, Julius Zint, Philip Matesanz, Torben Voltmer

Masterprojekt „Entwicklung und Analyse einer sicheren Web-Anwendung“
Wintersemester 18/19

28. November 2018

Inhaltsverzeichnis

1. Benutzte Methoden

1.1. Vier-Augen-Prinzip

Wir haben festgelegt, dass grundsätzlich jeder Code mindestens über das Vier-Augen-Prinzip abzusichern ist. Da wir hier ein sicheres System entwickeln wollen, versuchen wir dadurch das Risiko zu minimieren. Besonders kritische Bereiche oder Entscheidungen wurden als Gruppe, also durch Acht-Augen-Prinzip, in Angriff genommen.

1.2. Tests

Dadurch, dass wir das Play Framework vor dem Projekt nicht kannten, sowie dass wir einen agilen Entwicklungsprozess nutzen wollten, haben wir abgestimmt, möglichst viel Code durch Tests abzudecken. Hiermit wird ein eventuelles Refactoring einfacher und das Vertrauen in unsere Software wird verstärkt. In Anbetracht dessen, dass Änderungen an der Architektur beziehungsweise Implementation durch eine Bedrohungsanalyse respektive gefundene Sicherheitslücken ausgelöst werden können, und wir iterativ Funktionalität zu unserem System hinzufügen wollen, sind **automatisierbare** Tests unabdingbar. Um dieses Ziel für unsere funktionalen, aber auch nicht-funktionalen Anforderungen zu erreichen, benutzen wir die von Play mitgelieferte JUnit-Bibliothek.

1.2.1. Funktionale Tests

Zur Verifikation der funktionalen Korrektheit unseres System nutzen wir Unit-, seltener Integrations- aber auch Systemtests. Hierbei haben wir darauf geachtet, dass es uns durch „Dependency Injection“ möglich ist, die Abhängigkeiten einer Klasse durch „Mocks“ zu ersetzen. Dadurch wird garantiert, dass lediglich das Verhalten einer Klasse getestet wird, und dass dieses Verhalten nur auf dem durchs Interface vorgeschriebene Verhalten seiner Abhängigkeiten basiert.

Für die in dem Klassendiagramm (siehe Architekturbeschreibung **??**) aufgezeigten „Pakete“ werden folgende Voraussetzungen sowie Nachbedingungen getestet:

- Controller

  - Werden die Routen auf die richtigen, internen Methoden abgebildet?
  - Sind unsere Routen nur für authentisierte Nutzer erreichbar?
  - Kann die Login-Page ausschließlich ohne valide Session aufgerufen werden?
  - Geben die Methoden die erwarteten HTTP Status Codes zurück?
- Domänenlogik
  - Können nur die Nutzer mit passender Autorisierung die Operationen durchführen?
  - Verhalten sich die Methoden richtig, wenn nicht vorhandene Entity-IDs übergeben werden?
  - Entsprechen die Fehlermeldungen den erwarteten?
  - Kriege ich den richtigen Rückgabewert zurück?
  - Werden die reCAPTCHAs nach den richtigen Intervallen (5 pro Account, 50 pro IP-Adresse) angezeigt?
  - Ist eine IP-Adresse wirklich nach 100 fehlgeschlagenen Logins ausgesperrt?
  - Kann das Captcha in verschiedenen Browsern gelöst werden?
- Cross Cutting Concerns
  - Ist die Passwortlänge auf das von BCrypt vorgebene Limit gesetzt?

Bei den View-spezifischen „Data Transfer Object“-Klassen sehen wir keinen Bedarf zum Testen. Bei diesen handelt es sich nur um „Plain Old Data“-Objekte, die außer trivialen Zugriffsmethoden (Getter und Setter) keine weiteren Methoden besitzen. Dies gilt für alle trivialen Zugriffsmethoden unserer Klassen.
Darüber hinaus gibt es keine direkten Tests, die verifizieren, ob eine simple Relation von EBean sachgerecht auf das richtige Objekt abgebildet wird. Eine Ausnahme bilden hier die `User`-Entitäten, da beim Löschen eines Benutzers ebenfalls alle Gruppen gelöscht werden müssen, dessen Besitzer er ist. Für so eine kaskadierende Löschoperation wurde extra ein Test geschrieben.

Im Anhang befindet sich der Unittest Code Coverage Report D. Während sicherheitsrelevante Aspekte der Software wie z.B. die Policy oder die Login-Firewall eine sehr hohe Code Coverage aufweisen haben die Dtos eine sehr geringe. Diese Zahlen ergänzen sich mit der Zielsetzung eine sichere und gut funktionierende Software zu haben. Neben sicherheitsrelevanten Funktionen der Software erfahren auch die Manager-Klassen einen hohen Protzentwert. Die Code Coverage ermöglicht eine gute Übersicht über den Teststatus der Software, liefert aber keinen detaillierten Einblick welche sicherheitskritischen Aspekte durch diese abgedeckt sind. Diese Tests werden im nächsten Abschnitt im Detail betrachtet.

1.2.2. Tests zur Prüfung von sicherheitsrelevanten Funktionen

Um SQL Injections zu verhindern, machen wir uns die von „h2“ angebotene Option `ALLOW_LITERALS=NONE` zu eigen. Dazu haben wir 2 Tests geschrieben. Einer, der zunächst verifiziert, dass diese Option wirklich SQL Injections verhindert, wenn man sie bei einer beliebigen „h2“-Datenbank aktiviert. Ein weiterer Test prüft dann, ob unsere Applikation diese Option richtig gesetzt hat, damit wir sicher sein können, dass auch im Applikationskontext keine SQL Injections möglich sind.

Um garantieren zu können, dass unsere Autorisierung funktioniert, haben wir eine Reihe von Tests geschrieben, die mit verschiedenen Nutzern unterschiedlicher Privilegien versucht, die vorhandenen Operationen durchzuführen.

1.2.3. Penetrationstests

Funktionalität der CSRF Tokens

Um die Funktionalität der CSRF-Tokens sicherzustellen, wurde ein Penetrationstest durchgeführt. Im ersten Schritt wurde dazu einfach das CSRF-Token aus dem HTML des Formulars gelöscht. Diese Anfrage wird wie erwartet vom Server mit einer von Play vorgefertigten „`Unauthorized`“ Seite beantwortet. In einem weiteren Schritt wurde mit Curl eine Anfrage erstellt, in welcher lediglich Benutzername und Passwort, aber kein CSRF-Token enthalten war. Diese Anfrage wurde vom Server ohne Beanstandung akzeptiert. Der Grund hierfür ist, dass das CSRF-Token weder im Cookie noch als Formularwert an den Server übermittelt wird. Auch die `RequireCSRFCheck`-Annotation schafft hierfür keine Abhilfe. Die Möglichkeit für CSRF-Angriff besteht, wenn entweder beide der folgende Bedingungen gelten oder in abgeschwächter Form, wenn nur der zweite Punkt gegeben ist:

- Das Play-Session-Cookie wird nicht für die Session-Implementierung der Anwendung verwendet.
- Das Play-Session-Cookie ist nicht vorhanden.

In dieser Anwendung beinhaltet das Play-Session-Cookie alle Informationen zur Session eines Benutzers und somit ist lediglich beim Login der CSRF Schutz ausgehebelt, da zu diesem Zeitpunkt noch nicht zwangsläufig ein Play-Session-Cookie vorhanden ist.

Ein konkretes Angriffsszenario in dieser Anwendung könnte folgendermaßen aussehen: Das Opfer darf hierzu keinerlei Cookies von der HsH-Helfer Seite im Browser haben. Dies kann gegeben sein, wenn es seine Cookies gelöscht hat oder den „Incognito“-Modus im Browser verwendet. Ein Angreifer kann nun das Opfer auf eine von ihm präparierte Seite locken, wo durch das Klicken auf einen Link ein Login-Formular abgeschickt wird. Somit ist das Opfer angemeldet unter einem Benutzeraccount, welcher auch dem Angreifer bekannt ist. Ist das Opfer nun unvorsichtig und verifiziert den aktuellen Benutzeraccount

nicht, lädt es möglicherweise unter einem Benutzeraccount Dateien hoch, auf welchen der Angreifer Zugriff hat.

Das Angriffsrisiko erhöht sich, wenn die Session in einem eigenen Cookie hinterlegt wird. Ein konkretes Szenario hierfür wäre, wenn man für das Sessionkonzept eine Drittanbieter Library verwendet, die ihre Cookies selbst verwaltet. Ist dann das Session Cookie weiterhin vorhanden (läuft nicht ab) und das Play-Session-Cookie wurde nach einem Browser Neustart entfernt, ist beim ersten Request kein CSRF Schutz aktiv. Hierbei handelt es sich um reine Theorie, da diese Anwendung hier alle Informationen im Play-Session-Cookie hinterlegt.

CSRF-Angriff

Im Rahmen eines Penetrationstests wurde herausgefunden, dass die von Play generierten CSRF-Tokens keine Nonces darstellen beziehungsweise beliebig oft verwendet werden können. Sie werden mit einem in der Play-Session persistierten Wert abgeglichen und auf Authentizität geprüft. Dieser Vergleichswert wird dann gesetzt, wenn der Nutzer erstmalig eine Seite aufruft, die über ein Formular verfügt.

Es wurde herausgefunden, dass unser Session-Konzept diesen Wert nicht berücksichtigt: Meldet sich ein Nutzer an, wird der CSRF-Vergleichswert beim Login gesetzt, beim Logout jedoch nicht wieder entfernt. Bei einer Situation, bei der sich zwei Nutzer den gleichen physischen Computer beziehungsweise Browser teilen und sich nacheinander anmelden, würden beide Nutzer den gleichen CSRF-Vergleichswert verwenden. In der Folge könnte der erste Nutzer den Zweiten angreifen, da er valide CSRF-Tokens des Zweiten - basierend auf dem gleichen Vergleichswert - kennt.

In der Folge wurde das Session-Konzept so geändert, dass vor Anlegen einer neuen Session alle bisherigen Werte aus der korrospondierenden Play-Session entfernt werden: Bei einem Login ist so garantiert, dass ein neuer CSRF-Vergleichswert erstellt wird.

HTTP Response Splitting mittels Dateinamen

Um zu prüfen ob HTTP Response Splitting durch das Einfügen von Zeilenumbrüchen in Dateinamen möglich sind, wurde ein Penetrationstest durchgeführt. Dazu wurde vorübergehend die Eingabevalidierung für Dateinamen deaktiviert, um die Zeilenumbrüche platzieren zu können.

In der Oberfläche wird der Zeilenumbruch wie erwartet dargestellt. Beim Versuch die Datei herunterzaladen gibt es eine Exception: `[ServerResultException: Error converting Play Result for server backend]`. Der Benutzer bekommt eine Fehlerseite mit dem HTTP Statuscode 500 angezeigt und die Datei wird nicht heruntergeladen. Weiter Seiteneffekte konnten dabei nicht beobachtet werden. Dieses Verhalten wird durch das Play

Framework realisiert, dass die Verwendung von Zeilenumbrüchen in HTTP Headern unterbindet.

Das Fehlerverhalten ist hier angemessen. Druch die Eingabevalidierung ist es bei normaler Benutzung nicht möglich die Zeilenumbrüche in einem Dateinamen zu verwenden.

1.3. Reviews

1.3.1. Session-Konzept

Die erste Version unseres Session-Konzepts wurde einem Review unterzogen. Das Review wurde von einer Person innerhalb von 2 Stunden durchgeführt. Hierbei wurde insbesondere ein schlechtes Design bemängelt. Die Session-Funktionalität wurde nicht von einer beziehungsweise wenigen Klassen abgebildet, sondern auf verschiedenste Klassen verteilt: Die Login-Methode im Controller erstellte inline einen Datenbank-Eintrag sowie ein Cookie. Andere Controller-Methoden manipulierten inline die entsprechenden Einträge und entfernten nach Bedarf Cookies. Das Wissen darüber, was überhaupt eine Session darstellt war so nicht an einer zentralen Stelle im Code zu finden und entsprechend schwer zu prüfen.

Ebenfalls wurde ein Bug gefunden, der indirekt eine Folge des mangelhaften Designs war: Eine Endlosschleife von Redirects. Die Prüfung, ob für den aktuellen Benutzer eine „valide“ Session vorliegt, wurde von mehreren Klassen eigenständig implementiert, und fand auf unterschiedliche Art und Weise statt. Änderte sich während der Benutzung die Benutzer-IP, stellte dies für die eine Komponente eine valide Session dar, für die andere Komponente jedoch nicht. Die Controller-Methode leitete den Nutzer mangels „valider“ Session so zur Login Seite, diese stellte jedoch eine „valide“ Session fest und leitete den Nutzer erneut zur Controller-Methode.

Es wurde beschlossen, das Session-Konzept einem grundsätzlichen Rewrite zu unterziehen. Das Wissen darüber, was eine Session darstellt, sollte gebündelt an einer zentralen Stelle der Architektur liegen, der Rest der Anwendung nur die API dieser „zentralen Stelle“ verwenden und unter keinen Umständen selbst an Cookies oder den entsprechenden Datenbank-Entitäten manipulieren. Um sicherzustellen, dass es nicht erneut zu dem bereits festgestellten Redirect-Bug kommt, wurden diese Fälle durch Unit-Tests abgebildet.

1.3.2. Login Firewall

Die aktuelle Version unserer Login-Firewall wurde einem Review durch zwei Personen unterzogen, welches zwei Stunden benötigte. Hierbei wurde ein Bug entdeckt, der ein Informationsleck zur Folge hatte: Die Firewall versetzt Benutzeraccounts nach N

falschen Logins in einen Captcha-Modus. Login-Versuche auf nicht-existierende Benutzeraccounts, führten nicht zu dieser Account-„Sperre“. Aufgrund dieser funktional anderen Behandlung wäre es möglich gewesen, feststellen zu können, ob ein Account existiert oder nicht, was ein Brute-Forcing maßgeblich erleichtern würde.

Die Login-Firewall wurde angepasst und es wurde ein weiteres Review für die finale Version der Anwendung geplant, in dem geprüft werden soll, ob alle möglichen Login-Kombinationen für einen Außenstehenden „gleich“ sind, um ein solches Informationsleck zu verhindern.

1.3.3. Eingabevalidierung der HTML-Formulare

Um Daten aus übermittelten Formularen zu empfangen, nutzt unsere Anwendung DTOs, die mit Play-Constraint-Annotations versehen sind. Über diese Annotations wird sichergestellt, dass diese Daten einem bestimmten Format entsprechen, z.B. eine valide E-Mail Adresse darstellen. Diese Validierung wurde durch eine Person einem zweistündigen Review unterzogen.

Im Zuge des Reviews wurde festgestellt, dass das vorhandene Design Race Condition begünstigt. Die Validierungen im DTO waren nicht ausdrücklich auf „statische“ Tests begrenzt. Es wurden teilweise Validierungen vorgenommen, die Datenbankzugriffe beinhaltet haben, beispielsweise die Prüfung der Existenz eines Benutzernames. Wird im Controller die Formular-Validierung[1] aufgerufen, ist die im DTO vorgenommene Validierung sowie der Controller-Code nicht Teil einer Transaktion. D.h. wenn die Formular-Validation meldet, dass der Nutzername noch nicht existiert, ist dieser Zustand nicht mehr garantiert, wenn der Benutzer tatsächlich angelegt wird.

Auch wenn dies im vorliegenden Fall aufgrund der Unique-Constraints in der Datenbank kein Sicherheitsproblem darstellt, haben wir uns dazu entschlossen, von einem solchen Design abzusehen. Ein Design, was in derartiger Weise Race Conditions begünstigt, ist objektiv schlecht. In den DTOs finden inzwischen nur noch „statische“ Tests statt. Diese Entscheidung hat zur Entstehung der Manager-Klassen geführt. Diese bieten Methoden, die transaktionssicher sind und werfen beispielsweise eine Exception, wenn der Nutzer bereits existiert.

1.3.4. Bcrypt Password Hashing

Der Code zum sicheren Abspeichern eines Nutzerpassworts in der Datenbank wurde durch eine Person einem Review unterzogen. Die dabei aufgekommene Anregung wurde mit dem ganzen Team diskutiert. Das Passwort wird im Code von lediglich zwei Stellen aus gesetzt. Der „`UserManager`“ verwendet sie beim Zurücksetzen des Passworts und der „`LoginManager`“ beim Setzen eines neuen Passworts. In beiden Fällen wird direkt beim

[1] boundForm.hasErrors()

Aufruf des Setters der Hash durch die Verwendung des „`PasswordSecurityModule`“ erzeugt. Somit ist sichergestellt, dass beim aktuellen Codestand keine Klartextpasswörter in der Datenbank abgespeichert werden. BCrypt wird verwendet um den Hash zu generieren und es wurde kurz diskutiert ob man nicht doch auf PBKDF2 wechselt. Da beide aber nicht ohne eine Third-Party-Library auskommen, wurde davon abgesehen. Dem Bcrypt-Code [2] kann entnommen werden, dass das automatisch generierte Salt SecureRandom verwendet und somit ausreichend zufällige Salts für unterschiedliche Passwörter generiert und auch die Rundenanzahl, die explizit festgelegt wurde, ist ausreichend sicher.

Aus dem Review ging die Idee hervor, bei erfolgreichen Logins durch Messen der für das Hashing benötigten Zeit die Rundenanzahl dynamisch zu erhöhen, um somit für Zukunftssicherheit zu sorgen. Die Idee wurde ins Backlog aufgenommen und das Review damit abgeschlossen.

1.3.5. CSRF-Token Generierung und Prüfung von Play

Play verwendet zum Generieren von CSRF-Tokens die Java Klasse SecureRandom als CPRNG:

```
def generateToken: String = {
  val bytes = new Array[Byte](12)
  random.nextBytes(bytes)
  new String(Hex.encodeHex(bytes))
}
```

Quellcode 1.1: Methode zum Generieren von Tokens. Aus DefaultCSRFTokenSigner, Play 2.6.20

Dieser Token wird allerdings nicht direkt verwendet. Stattdessen wird er mit einer Nonce konkateniert. Das Ergebnis wird anschließend mittels HMAC-SHA1 und einem privaten Schlüssel signiert.

```
def signToken(token: String): String = {
  val nonce = clock.millis()
  val joined = nonce + "-" + token
  signer.sign(joined) + "-" + joined
}
```

Quellcode 1.2: Methode zum Signieren von Token. Aus DefaultCSRFTokenSigner, Play 2.6.20

Das Format eines Play CSRF Token ist demnach

```
<signature> - <nonce> - <token>
```

Für eine Play-Session wird nur ein Token generiert (aus Listing 1.1). Dieses ist solange gültig wie die Play-Session und ändert sich nicht. Es wird für jedes Formular innerhalb einer Session verwendet. Da die Nonce auf der Systemzeit basiert, ändert sich diese jedoch mit jedem Request, was zu unterschiedlichen Signaturen führt. Dadurch werden Attacken, die ähnlich wie `BREACH` die HTTP-Komprimierung bei Verwendung von HTTPS angreift, verhindert[2].

Um CSRF-Token überprüfen zu können, wird das erste CSRF-Token, dass in einer Play Session verwendet wird im Play-Session-Cookie gespeichert. Durch einen Vergleich des Tokens im ersten CSRF-Token mit dem zu prüfenden Token kann sichergestellt werden, dass das CSRF-Token zu der richtigen Session gehört. Außerdem wird geprüft ob die Signatur eines CSRF-Tokens korrekt ist.

1.4. Statische Codeanalyse

Da innerhalb der Gruppe keine Präferenzen bezüglich eines Werkzeugs für die statische Codeanalyse bestand, haben wir verschiedene ausprobiert. Die Erfahrung aus anderen Sprachökosystemen (beispielsweise C++) zeigt, dass es dennoch ratsam ist, unterschiedliche Werkzeuge zu verwenden, um eine breitere Menge abzudecken und gegebenfalls „False Positives“ auszuschließen.

Die Werkzeuge, die wir nach einer kurzen Evaluation ausgesucht haben, sind:

- FindBugs [3] inkl. „find-sec-bugs“-Plugin [1],
- Synopsis SecureAssist (ehemalig Cigital SecureAssist) [5],
- SonarLint (ehemalig Sonarqube) [4]

Allerdings stoßen diese Werkzeuge hier an ihre Grenzen, da sie den autogenerierten Code des Play Frameworks, Play-spezifischen Code sowie Scala nicht korrekt analysieren konnten. Aus diesem Grund führen wir die gefundenen Mängel auf, evaluieren deren Gültigkeit, werden aber folgend aufgrund des enttäuschenden Ergebnisses diese Werkzeuge nicht mehr verwenden.

[2] Siehe dazu https://www.playframework.com/documentation/2.6.x/JavaCsrf unten

1.4.1. FindBugs

Wir haben die Version 3.0.1 verwendet, die am 2018-10-16 veröffentlicht wurde. Bei dem Plugin „find-sec-bugs“ wurde Version 1.8 verwendet.

Verwendete Einstellungen:

- Analyseaufwand: Maximal
- Minimalster Fehlerrang: 20 - Of Concern
- Alle Fehlerkategorien, Filter wurden aktiviert

Da ein Großteil der Fehler, die „FindBugs“ gefunden hat, sich in den Tests, im autogenerierten Code oder den Templates vom Play Framework befindet, war es schwer, die „False Positives“ herauszufiltern. Eine gesamte Auflistung aller gefundenen Fehler kann im Anhang C eingesehen werden. Gefunden wurden Fehler in den Kategorien „Bad Practice“, „Richtigkeit“, „Performance“, „Dodgy Code“ sowie „Sicherheitsanfälligkeit durch bösartigen Code“. In unserem eigentlichen Applikationscode wurden nur Fehler der Kategorie „Sicherheitsanfälligkeit durch bösartigen Code“ gefunden, die darauf hinweisen, dass einige Variablen mit „final“ deklariert werden sollten.

1.4.2. Synopsis SecureAssist

Wir haben die Version 3.3.0 verwendet, welche am 2018-01-29 veröffentlicht wurde. Da dieses Werkzeug in Betracht auf Sicherheitslücken von berühmten Sicherheitsexperten wie Gary McGraw entwickelt wurde, hoffen wir, dass dieses Werkzeug Lücken findet, die den anderen Werkzeugen nicht auffallen.

Hardkodierte Passwörter

In unserer Testklasse `UserManagerTest` verwenden wir ein hardkodiertes Passwort, um zu überprüfen, ob während des „Passwort zurücksetzen“-Prozeses für einen Nutzer ein temporäres Passwort erstellt werden kann, zu sehen in Quellcode A.1. Das sehen wir nicht als sicherheitsrelevanten Fehler an und ignorieren diesen daher.

Query Injections

Die Regel, die SecureAssist verwendet, um mögliche SQL Injections zu finden, scheint nur zu prüfen, ob ein SQL Statement als String verwendet wird, und ob es als `PreparedStatement` verwendet wird. Leider überprüft es nicht, ob das als String angegebene SQL

Statement überhaupt veränderbar ist; beispielsweise durch Konkatenieren eines Parameters. Das Werkzeug hat daher ein „False Positive“ in unserer `DatabaseInitialization`-Klasse gefunden, siehe Quellcode B.1. Die Statements in dieser Klasse können nicht von einem Nutzer durch Parameter angereichert werden. Daher ignorieren wir diesen Mangel.

Informationsleck

Das Ausgeben des Stacktrace im Falle einer Exception sieht das Werkzeug bereits als Verletzung der Vertraulichkeit, welchem wir zustimmen. Glücklicherweise schützt Play uns in diesem Fall und gibt keine Stacktraces an einen Benutzer weiter, sofern die Applikation im „Production Mode“ ausgeführt wird. In diesem Fall wird die Exception mit einer ID ausgezeichnet, die ID wird dem Benutzer angezeigt und der Stacktrace wird in das Applikationslog geschrieben.

1.4.3. SonarLint

An dem „SonarLint“-Plugin für die IntelliJ IDE konnten keine relevanten Einstellungen verändert werden, daher wurden die Werkseinstellungen benutzt.

Die Fehler, die während des Scans unseres Quellcodes gefunden wurden, werden hier der Vollständigkeit halber aufgeführt, obwohl nur programmierstil-spezifische Fehler gefunden wurden.

Verbergen des `public`-Konstruktors

Die Klassen

- `ConstraintValues`,
- `Authentification`,
- `LaggyDT`,
- `Recaptcha`

sind entweder reine Utilityklassen mit statischen Methoden beziehungsweise finalen Variabeln oder implementieren eine Annotation. Es bietet sich also an, einen `private`-Konstruktor für diese Klassen zu schreiben, damit der `public`-Konstruktor verborgen ist, und niemand ausversehen ein Objekt dieser Klassen instanziieren kann.

Extrahieren von Konstanten in `final`-Variabeln

In der `Firewall` sowie unseren `Manager`-Klassen verwenden wir Konstanten, die an mehreren Stellen verwendet werden. Im Falle der `Manager`s handelt es sich um ähnliche String, die in einer Exception den Fehler genauer beschreiben. Bei diesen Strings könnte man sich der `i18n`-Implementation von Play bedienen, um diese Fehlermeldungen für alle `Manager`s erreichbar zu machen, aber da Internationalisierung nicht im Umfang dieses Projekts ist, werden sie als statische, finale Variabeln extrahiert.

2. Gefundene Fehler

In der folgenden Tabelle stellen wir dar, welche Fehler durch welche Methode gefunden wurden.

Tabelle 2.1.: Auflistung der Fehler und Methode, die diese gefunden hat

Fehler	**Methode**	**Zeit (std.)**
reCAPTCHA nicht lösbar in Firefox	Manueller Test	1
`NullPointerException` beim Validieren des „Passwort nach Zurücksetzung ändern“-Formulars	Funktionaler Test der Controllermethode	0.5
Löschen eines Nutzers führte nicht zum Löschen seiner Gruppe	Funktionaler Test der Controllermethode	0.5
Bootstrap-Helper-Methoden zur Erstellung eines Formulars haben Felder nicht mit bestehenden Daten befüllt	Aufgefallen während eines Reviews	0.25
„Nutzer erstellen“-Prozess konnte von einem „Nicht-Administrator“ ausgeführt werden	Autorisierungstests	0.5
Mögliche Race Conditions während der Validation eines Nutzernamens	Review	0.5
Mögliche Side-Channel-Attacke beim Loginversuch durch den Login-Firewall	Review	0.75
Endlosschleife von Redirects beim Login eines Nutzers mit valider Session aber unterschiedlicher IP	Review	1.5

3. Literatur

[1] Philipp Arteau. *Find Security Bugs - The FindBugs plugin for security audits of Java web applications*. URL: https://find-sec-bugs.github.io/ (besucht am 03. 11. 2018).

[2] *Bcrypt Quellcode*. URL: https://github.com/jeremyh/jBCrypt/blob/master/src/main/java/org/mindrot/BCrypt.java (besucht am 10. 11. 2018).

[3] *FindBugs - Find bugs in Java Programs*. URL: https://github.com/findbugsproject/findbugs (besucht am 03. 11. 2018).

[4] *SonarLint - Fix issues before they exist*. URL: https://www.sonarlint.org/ (besucht am 03. 11. 2018).

[5] *Synopsis SecureAssist*. URL: https://www.synopsys.com/software-integrity/resources/datasheets/secureassist.html (besucht am 03. 11. 2018).

Appendix

A. src/test/domainlogic/usermanager/UserManagerTest.java

Quellcode A.1: Hardkodiertes Passwort in einem Test aus der Klasse `UserManagerTest`

```
@Test
public void testChangePassword() throws InvalidArgumentException {
    String testUsername = "test";
    String newPassword = "0123456789";
}
```

B. src/app/DatabaseInitialization.java

Quellcode B.1: Von SecureAssist gefundenes False Positive Beispiel für Query Injections

```
db.withConnection(connection -> {
  Statement stmt = connection.createStatement();
  stmt.execute("SET␣REFERENTIAL_INTEGRITY␣FALSE");
  stmt.execute("TRUNCATE␣TABLE␣groupmembers");
  stmt.execute("TRUNCATE␣TABLE␣users");
  stmt.execute("TRUNCATE␣TABLE␣groups");
  stmt.execute("TRUNCATE␣TABLE␣internal_session");
  stmt.execute("SET␣REFERENTIAL_INTEGRITY␣TRUE");
  stmt.execute("SET␣ALLOW_LITERALS␣NONE");
});
```

C. FindBugs Report

FindBugs Report

FindBugs Report

Produced using FindBugs3.0.1.

Project: HshHelper[HshHelper]

Metrics

5478 lines of code analyzed, in 144 classes, in 21 packages.

Metric	Total	Density*
High Priority Warnings	11	2.01
Medium Priority Warnings	102	18.62
Total Warnings	**113**	**20.63**

(Defects per Thousand lines of non-commenting source statements)*

Summary

Warning Type	Number
Bad practice Warnings	38
Correctness Warnings	1
Experimental Warnings	1
Malicious code vulnerability Warnings	61
Performance Warnings	1
Dodgy code Warnings	11
Total	**113**

Warnings

Click on each warning link to see a full description of the issue, and details of how to resolve it.

Bad practice Warnings

Warning	Priority	Details
Class names should start with an upper case letter	Medium	The class name controllers.routes doesn't start with an upper case letter In file routes.java, lines 9 to 15 In class controllers.routes At routes.java:[lines 9-15]
Class names should start with an upper case letter	Medium	The class name controllers.routes$javascript doesn't start with an upper case letter In file routes.java, lines 17 to 23 In class controllers.routes$javascript At routes.java:[lines 17-23]
Method names should start with a lower case letter	Medium	The method name domainlogic.loginmanager.Authentification.Perform(String, String) doesn't start with a lower case letter In file Authentification.java, lines 35 to 47 In class domainlogic.loginmanager.Authentification In method domainlogic.loginmanager.Authentification.Perform(String, String) At Authentification.java:[lines 35-47]

Method names should start with a lower case letter	Medium	The method name extension.RecaptchaHelper.CaptchaField() doesn't start with a lower case letter In file RecaptchaHelper.java, lines 16 to 17 In class extension.RecaptchaHelper In method extension.RecaptchaHelper.CaptchaField() At RecaptchaHelper.java:[lines 16-17]
Method names should start with a lower case letter	Medium	The method name extension.RecaptchaHelper.IsValidResponse(String, String) doesn't start with a lower case letter In file RecaptchaHelper.java, lines 28 to 43 In class extension.RecaptchaHelper In method extension.RecaptchaHelper.IsValidResponse(String, String) At RecaptchaHelper.java:[lines 28-43]
Field names should start with a lower case letter	Medium	The field name policy.ext.loginFirewall.Firewall.LaggyTsIntervalHours doesn't start with a lower case letter In file Firewall.java, lines to In class policy.ext.loginFirewall.Firewall Field policy.ext.loginFirewall.Firewall.LaggyTsIntervalHours In Firewall.java
Field names should start with a lower case letter	Medium	The field name policy.ext.loginFirewall.Firewall.RelevantHours doesn't start with a lower case letter In file Firewall.java, lines to In class policy.ext.loginFirewall.Firewall Field policy.ext.loginFirewall.Firewall.RelevantHours In Firewall.java
Method names should start with a lower case letter	Medium	The method name policy.ext.loginFirewall.LaggyDT.Get() doesn't start with a lower case letter In file LaggyDT.java, lines 8 to 12 In class policy.ext.loginFirewall.LaggyDT In method policy.ext.loginFirewall.LaggyDT.Get() At LaggyDT.java:[lines 8-12]
Method names should start with a lower case letter	Medium	The method name policy.session.SessionManager.SessionInstance() doesn't start with a lower case letter In file SessionManager.java, line 21 In class policy.session.SessionManager In method policy.session.SessionManager.SessionInstance() At SessionManager.java:[line 21]
Method names should start with a lower case letter	Medium	The method name views.bootstrapHelper.enums.ButtonType$.Button() doesn't start with a lower case letter In file ButtonEnum.scala, line 4 In class views.bootstrapHelper.enums.ButtonType$ In method views.bootstrapHelper.enums.ButtonType$.Button() At ButtonEnum.scala:[line 4]
		The method name views.bootstrapHelper.enums.ButtonType$.Submit()

Method names should start with a lower case letter	Medium	doesn't start with a lower case letter In file ButtonEnum.scala, line 5 In class views.bootstrapHelper.enums.ButtonType$ In method views.bootstrapHelper.enums.ButtonType$.Submit() At ButtonEnum.scala:[line 5]
Method names should start with a lower case letter	Medium	The method name views.bootstrapHelper.enums.ButtonType$.Type() doesn't start with a lower case letter In file ButtonEnum.scala, line 6 In class views.bootstrapHelper.enums.ButtonType$ In method views.bootstrapHelper.enums.ButtonType$.Type() At ButtonEnum.scala:[line 6]
Method names should start with a lower case letter	Medium	The method name views.bootstrapHelper.enums.LinkClass$.Primary() doesn't start with a lower case letter In file LinkEnum.scala, line 4 In class views.bootstrapHelper.enums.LinkClass$ In method views.bootstrapHelper.enums.LinkClass$.Primary() At LinkEnum.scala:[line 4]
Method names should start with a lower case letter	Medium	The method name views.bootstrapHelper.enums.LinkClass$.Secondary() doesn't start with a lower case letter In file LinkEnum.scala, line 5 In class views.bootstrapHelper.enums.LinkClass$ In method views.bootstrapHelper.enums.LinkClass$.Secondary() At LinkEnum.scala:[line 5]
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.button doesn't start with an upper case letter In file button.template.scala, lines to In class views.html.bootstrapHelper.button In button.template.scala
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.button$ doesn't start with an upper case letter In file button.template.scala, lines 29 to 58 In class views.html.bootstrapHelper.button$ At button.template.scala:[lines 29-58]
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.card doesn't start with an upper case letter In file card.template.scala, lines to In class views.html.bootstrapHelper.card In card.template.scala
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.card$ doesn't start with an upper case letter In file card.template.scala, lines 27 to 47 In class views.html.bootstrapHelper.card$ At card.template.scala:[lines 27-47]
		The class name views.html.bootstrapHelper.cardWithTitle doesn't start with an upper case letter

Class names should start with an upper case letter	Medium	In file cardWithTitle.template.scala, lines to In class views.html.bootstrapHelper.cardWithTitle In cardWithTitle.template.scala
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.cardWithTitle$ doesn't start with an upper case letter In file cardWithTitle.template.scala, lines 27 to 50 In class views.html.bootstrapHelper.cardWithTitle$ At cardWithTitle.template.scala:[lines 27-50]
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.center doesn't start with an upper case letter In file center.template.scala, lines to In class views.html.bootstrapHelper.center In center.template.scala
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.center$ doesn't start with an upper case letter In file center.template.scala, lines 27 to 52 In class views.html.bootstrapHelper.center$ At center.template.scala:[lines 27-52]
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.form doesn't start with an upper case letter In file form.template.scala, lines to In class views.html.bootstrapHelper.form In form.template.scala
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.form$ doesn't start with an upper case letter In file form.template.scala, lines 27 to 92 In class views.html.bootstrapHelper.form$ At form.template.scala:[lines 27-92]
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.link doesn't start with an upper case letter In file link.template.scala, lines to In class views.html.bootstrapHelper.link In link.template.scala
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.link$ doesn't start with an upper case letter In file link.template.scala, lines 28 to 48 In class views.html.bootstrapHelper.link$ At link.template.scala:[lines 28-48]
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.listGroupButton doesn't start with an upper case letter In file listGroupButton.template.scala, lines to In class views.html.bootstrapHelper.listGroupButton In listGroupButton.template.scala The class name views.html.bootstrapHelper.listGroupButton$ doesn't start with an upper case letter

Class names should start with an upper case letter	Medium	In file listGroupButton.template.scala, lines 29 to 63 In class views.html.bootstrapHelper.listGroupButton$ At listGroupButton.template.scala:[lines 29-63]
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.listGroupFlush doesn't start with an upper case letter In file listGroupFlush.template.scala, lines to In class views.html.bootstrapHelper.listGroupFlush In listGroupFlush.template.scala
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.listGroupFlush$ doesn't start with an upper case letter In file listGroupFlush.template.scala, lines 27 to 55 In class views.html.bootstrapHelper.listGroupFlush$ At listGroupFlush.template.scala:[lines 27-55]
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.listGroupLink doesn't start with an upper case letter In file listGroupLink.template.scala, lines to In class views.html.bootstrapHelper.listGroupLink In listGroupLink.template.scala
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.listGroupLink$ doesn't start with an upper case letter In file listGroupLink.template.scala, lines 29 to 49 In class views.html.bootstrapHelper.listGroupLink$ At listGroupLink.template.scala:[lines 29-49]
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.mainContent doesn't start with an upper case letter In file mainContent.template.scala, lines to In class views.html.bootstrapHelper.mainContent In mainContent.template.scala
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.mainContent$ doesn't start with an upper case letter In file mainContent.template.scala, lines 27 to 53 In class views.html.bootstrapHelper.mainContent$ At mainContent.template.scala:[lines 27-53]
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.sidebarPlusContent doesn't start with an upper case letter In file sidebarPlusContent.template.scala, lines to In class views.html.bootstrapHelper.sidebarPlusContent In sidebarPlusContent.template.scala
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.sidebarPlusContent$ doesn't start with an upper case letter In file sidebarPlusContent.template.scala, lines 27 to 54 In class views.html.bootstrapHelper.sidebarPlusContent$ At sidebarPlusContent.template.scala:[lines 27-54]
		The class name views.html.bootstrapHelper.table doesn't

Class names should start with an upper case letter	Medium	start with an upper case letter In file table.template.scala, lines to In class views.html.bootstrapHelper.table In table.template.scala
Class names should start with an upper case letter	Medium	The class name views.html.bootstrapHelper.table$ doesn't start with an upper case letter In file table.template.scala, lines 27 to 123 In class views.html.bootstrapHelper.table$ At table.template.scala:[lines 27-123]

Correctness Warnings

Warning	Priority	Details
Null pointer dereference	Medium	Null pointer dereference of ? in new router.Routes$$anonfun$routes$1(Routes) In file Routes.scala, line 480 In class router.Routes$$anonfun$routes$1 In method new router.Routes$$anonfun$routes$1(Routes) Local variable stored in JVM register ? Dereferenced at Routes.scala:[line 480]

Experimental Warnings

Warning	Priority	Details
Method may fail to clean up stream or resource	Medium	DatabaseInitialization.lambdanew0(Connection) may fail to clean up java.sql.Statement In file DatabaseInitialization.java, line 26 In class DatabaseInitialization In method DatabaseInitialization.lambdanew0(Connection) Reference type java.sql.Statement 1 instances of obligation remaining Obligation to clean up resource created at DatabaseInitialization.java:[line 26] is not discharged Path continues at DatabaseInitialization.java:[line 27] Path continues at DatabaseInitialization.java:[line 28] Path continues at DatabaseInitialization.java:[line 29] Path continues at DatabaseInitialization.java:[line 30] Path continues at DatabaseInitialization.java:[line 31] Path continues at DatabaseInitialization.java:[line 32] Path continues at DatabaseInitialization.java:[line 33] Remaining obligations: {Statement x 1}

Malicious code vulnerability Warnings

Warning	Priority	Details
		domainlogic.groupmanager.GroupManagerTests.adminId isn't final

		but should be
Field isn't final but should be	High	In file GroupManagerTests.java, line 47 In class domainlogic.groupmanager.GroupManagerTests Field domainlogic.groupmanager.GroupManagerTests.adminId At GroupManagerTests.java:[line 47]
		domainlogic.groupmanager.GroupManagerTests.adminsGrpId isn't final but should be
Field isn't final but should be	High	In file GroupManagerTests.java, line 56 In class domainlogic.groupmanager.GroupManagerTests Field domainlogic.groupmanager.GroupManagerTests.adminsGrpId At GroupManagerTests.java:[line 56]
		domainlogic.groupmanager.GroupManagerTests.allId isn't final but should be
Field isn't final but should be	High	In file GroupManagerTests.java, line 54 In class domainlogic.groupmanager.GroupManagerTests Field domainlogic.groupmanager.GroupManagerTests.allId At GroupManagerTests.java:[line 54]
		domainlogic.groupmanager.GroupManagerTests.klausId isn't final but should be
Field isn't final but should be	High	In file GroupManagerTests.java, line 51 In class domainlogic.groupmanager.GroupManagerTests Field domainlogic.groupmanager.GroupManagerTests.klausId At GroupManagerTests.java:[line 51]
		domainlogic.groupmanager.GroupManagerTests.peterId isn't final but should be
Field isn't final but should be	High	In file GroupManagerTests.java, line 49 In class domainlogic.groupmanager.GroupManagerTests Field domainlogic.groupmanager.GroupManagerTests.peterId At GroupManagerTests.java:[line 49]
		domainlogic.groupmanager.GroupManagerTests.petersGrpId isn't final but should be
Field isn't final but should be	High	In file GroupManagerTests.java, line 58 In class domainlogic.groupmanager.GroupManagerTests Field domainlogic.groupmanager.GroupManagerTests.petersGrpId At GroupManagerTests.java:[line 58]
		domainlogic.loginmanager.LoginManagerTest.hashHelper isn't final but should be
Field isn't final but should be	High	In file LoginManagerTest.java, line 28 In class domainlogic.loginmanager.LoginManagerTest Field domainlogic.loginmanager.LoginManagerTest.hashHelper At LoginManagerTest.java:[line 28]
		policy.session.InternalSession.finder isn't final but should be
Field isn't final but should be	High	In file InternalSession.java, line 61 In class policy.session.InternalSession Field policy.session.InternalSession.finder At InternalSession.java:[line 61]
		policy.session.SessionManagerTest.defaultIp isn't final but should be
Field isn't final but should be	High	In file SessionManagerTest.java, line 23 In class policy.session.SessionManagerTest

		Field policy.session.SessionManagerTest.defaultIp At SessionManagerTest.java:[line 23]
		policy.session.SessionManagerTest.otherIp isn't final but should be
Field isn't final but should be	High	In file SessionManagerTest.java, line 24 In class policy.session.SessionManagerTest Field policy.session.SessionManagerTest.otherIp At SessionManagerTest.java:[line 24]
		policy.Specification.instance isn't final but should be
Field isn't final but should be	High	In file Specification.java, line 8 In class policy.Specification Field policy.Specification.instance At Specification.java:[line 8]
		domainlogic.groupmanager.GroupManagerTests.admin should be package protected
Field should be package protected	Medium	In file GroupManagerTests.java, lines to In class domainlogic.groupmanager.GroupManagerTests Field domainlogic.groupmanager.GroupManagerTests.admin In GroupManagerTests.java
		domainlogic.groupmanager.GroupManagerTests.admins should be package protected
Field should be package protected	Medium	In file GroupManagerTests.java, lines to In class domainlogic.groupmanager.GroupManagerTests Field domainlogic.groupmanager.GroupManagerTests.admins In GroupManagerTests.java
		domainlogic.groupmanager.GroupManagerTests.all should be package protected
Field should be package protected	Medium	In file GroupManagerTests.java, lines to In class domainlogic.groupmanager.GroupManagerTests Field domainlogic.groupmanager.GroupManagerTests.all In GroupManagerTests.java
		domainlogic.groupmanager.GroupManagerTests.klaus should be package protected
Field should be package protected	Medium	In file GroupManagerTests.java, lines to In class domainlogic.groupmanager.GroupManagerTests Field domainlogic.groupmanager.GroupManagerTests.klaus In GroupManagerTests.java
		domainlogic.groupmanager.GroupManagerTests.peter should be package protected
Field should be package protected	Medium	In file GroupManagerTests.java, lines to In class domainlogic.groupmanager.GroupManagerTests Field domainlogic.groupmanager.GroupManagerTests.peter In GroupManagerTests.java
		domainlogic.groupmanager.GroupManagerTests.petersGroup should be package protected
Field should be package protected	Medium	In file GroupManagerTests.java, lines to In class domainlogic.groupmanager.GroupManagerTests Field domainlogic.groupmanager.GroupManagerTests.petersGroup In GroupManagerTests.java
		domainlogic.loginmanager.LoginManagerTest.annika should be

		package protected
Field should be package protected	Medium	In file LoginManagerTest.java, lines to In class domainlogic.loginmanager.LoginManagerTest Field domainlogic.loginmanager.LoginManagerTest.annika In LoginManagerTest.java
		domainlogic.loginmanager.LoginManagerTest.app should be package protected
Field should be package protected	Medium	In file LoginManagerTest.java, lines to In class domainlogic.loginmanager.LoginManagerTest Field domainlogic.loginmanager.LoginManagerTest.app In LoginManagerTest.java
		domainlogic.loginmanager.LoginManagerTest.lydia should be package protected
Field should be package protected	Medium	In file LoginManagerTest.java, lines to In class domainlogic.loginmanager.LoginManagerTest Field domainlogic.loginmanager.LoginManagerTest.lydia In LoginManagerTest.java
		LoginFirewallTests.app should be package protected
Field should be package protected	Medium	In file LoginFirewallTests.java, lines to In class LoginFirewallTests Field LoginFirewallTests.app In LoginFirewallTests.java
		policy.ext.loginFirewall.Firewall.LaggyTsIntervalHours isn't final but should be
Field isn't final but should be	Medium	In file Firewall.java, line 12 In class policy.ext.loginFirewall.Firewall Field policy.ext.loginFirewall.Firewall.LaggyTsIntervalHours At Firewall.java:[line 12]
		policy.ext.loginFirewall.Firewall.NIPLoginsTriggerBan isn't final but should be
Field isn't final but should be	Medium	In file Firewall.java, line 15 In class policy.ext.loginFirewall.Firewall Field policy.ext.loginFirewall.Firewall.NIPLoginsTriggerBan At Firewall.java:[line 15]
		policy.ext.loginFirewall.Firewall.NIPLoginsTriggerVerification isn't final but should be
Field isn't final but should be	Medium	In file Firewall.java, line 14 In class policy.ext.loginFirewall.Firewall Field policy.ext.loginFirewall.Firewall.NIPLoginsTriggerVerification At Firewall.java:[line 14]
		policy.ext.loginFirewall.Firewall.NUidLoginsTriggerVerification isn't final but should be
Field isn't final but should be	Medium	In file Firewall.java, line 16 In class policy.ext.loginFirewall.Firewall Field policy.ext.loginFirewall.Firewall.NUidLoginsTriggerVerification At Firewall.java:[line 16]
		policy.ext.loginFirewall.Firewall.RelevantHours isn't final but should be

Field isn't final but should be	Medium	In file Firewall.java, line 13 In class policy.ext.loginFirewall.Firewall Field policy.ext.loginFirewall.Firewall.RelevantHours At Firewall.java:[line 13]
Field should be package protected	Medium	policy.session.SessionManagerTest.app should be package protected In file SessionManagerTest.java, lines to In class policy.session.SessionManagerTest Field policy.session.SessionManagerTest.app In SessionManagerTest.java
Field should be package protected	Medium	policy.session.SessionManagerTest.hashHelper should be package protected In file SessionManagerTest.java, lines to In class policy.session.SessionManagerTest Field policy.session.SessionManagerTest.hashHelper In SessionManagerTest.java
Field should be package protected	Medium	policy.session.SessionManagerTest.robin should be package protected In file SessionManagerTest.java, lines to In class policy.session.SessionManagerTest Field policy.session.SessionManagerTest.robin In SessionManagerTest.java
Field should be package protected	Medium	policy.session.SessionManagerTest.sessionManager should be package protected In file SessionManagerTest.java, lines to In class policy.session.SessionManagerTest Field policy.session.SessionManagerTest.sessionManager In SessionManagerTest.java
Field should be package protected	Medium	router.RoutesPrefix$.MODULE$ should be package protected In file RoutesPrefix.scala, lines to In class router.RoutesPrefix$ Field router.RoutesPrefix$.MODULE$ In RoutesPrefix.scala
Field isn't final and can't be protected from malicious code	Medium	views.bootstrapHelper.enums.ButtonClass$.MODULE$ isn't final and can't be protected from malicious code In file ButtonEnum.scala, lines to In class views.bootstrapHelper.enums.ButtonClass$ Field views.bootstrapHelper.enums.ButtonClass$.MODULE$ In ButtonEnum.scala
Field isn't final and can't be protected from malicious code	Medium	views.bootstrapHelper.enums.ButtonType$.MODULE$ isn't final and can't be protected from malicious code In file ButtonEnum.scala, lines to In class views.bootstrapHelper.enums.ButtonType$ Field views.bootstrapHelper.enums.ButtonType$.MODULE$ In ButtonEnum.scala
Field isn't final and can't be protected from malicious code	Medium	views.bootstrapHelper.enums.InputType$.MODULE$ isn't final and can't be protected from malicious code In file InputEnum.scala, lines to In class views.bootstrapHelper.enums.InputType$ Field views.bootstrapHelper.enums.InputType$.MODULE$ In InputEnum.scala

		views.bootstrapHelper.enums.LinkClass$.MODULE$ isn't final and can't be protected from malicious code
Field isn't final and can't be protected from malicious code	Medium	In file LinkEnum.scala, lines to In class views.bootstrapHelper.enums.LinkClass$ Field views.bootstrapHelper.enums.LinkClass$.MODULE$ In LinkEnum.scala
		views.bootstrapHelper.Input$.MODULE$ isn't final and can't be protected from malicious code
Field isn't final and can't be protected from malicious code	Medium	In file Input.scala, lines to In class views.bootstrapHelper.Input$ Field views.bootstrapHelper.Input$.MODULE$ In Input.scala
		views.html.bootstrapHelper.BootstrapFieldConstructor$.MODULE$ should be package protected
Field should be package protected	Medium	In file BootstrapFieldConstructor.template.scala, lines to In class views.html.bootstrapHelper.BootstrapFieldConstructor$ Field views.html.bootstrapHelper.BootstrapFieldConstructor$.MODULE$ In BootstrapFieldConstructor.template.scala
		views.html.bootstrapHelper.button$.MODULE$ isn't final and can't be protected from malicious code
Field isn't final and can't be protected from malicious code	Medium	In file button.template.scala, lines to In class views.html.bootstrapHelper.button$ Field views.html.bootstrapHelper.button$.MODULE$ In button.template.scala
		views.html.bootstrapHelper.card$.MODULE$ isn't final and can't be protected from malicious code
Field isn't final and can't be protected from malicious code	Medium	In file card.template.scala, lines to In class views.html.bootstrapHelper.card$ Field views.html.bootstrapHelper.card$.MODULE$ In card.template.scala
		views.html.bootstrapHelper.cardWithTitle$.MODULE$ isn't final and can't be protected from malicious code
Field isn't final and can't be protected from malicious code	Medium	In file cardWithTitle.template.scala, lines to In class views.html.bootstrapHelper.cardWithTitle$ Field views.html.bootstrapHelper.cardWithTitle$.MODULE$ In cardWithTitle.template.scala
		views.html.bootstrapHelper.center$.MODULE$ isn't final and can't be protected from malicious code
Field isn't final and can't be protected from malicious code	Medium	In file center.template.scala, lines to In class views.html.bootstrapHelper.center$ Field views.html.bootstrapHelper.center$.MODULE$ In center.template.scala
		views.html.bootstrapHelper.form$.MODULE$ isn't final and can't be protected from malicious code
Field isn't final and can't be protected from malicious code	Medium	In file form.template.scala, lines to In class views.html.bootstrapHelper.form$ Field views.html.bootstrapHelper.form$.MODULE$ In form.template.scala

		views.html.bootstrapHelper.link$.MODULE$ isn't final and can't be protected from malicious code
Field isn't final and can't be protected from malicious code	Medium	In file link.template.scala, lines to In class views.html.bootstrapHelper.link$ Field views.html.bootstrapHelper.link$.MODULE$ In link.template.scala
		views.html.bootstrapHelper.listGroupButton$.MODULE$ isn't final and can't be protected from malicious code
Field isn't final and can't be protected from malicious code	Medium	In file listGroupButton.template.scala, lines to In class views.html.bootstrapHelper.listGroupButton$ Field views.html.bootstrapHelper.listGroupButton$.MODULE$ In listGroupButton.template.scala
		views.html.bootstrapHelper.listGroupFlush$.MODULE$ isn't final and can't be protected from malicious code
Field isn't final and can't be protected from malicious code	Medium	In file listGroupFlush.template.scala, lines to In class views.html.bootstrapHelper.listGroupFlush$ Field views.html.bootstrapHelper.listGroupFlush$.MODULE$ In listGroupFlush.template.scala
		views.html.bootstrapHelper.listGroupLink$.MODULE$ isn't final and can't be protected from malicious code
Field isn't final and can't be protected from malicious code	Medium	In file listGroupLink.template.scala, lines to In class views.html.bootstrapHelper.listGroupLink$ Field views.html.bootstrapHelper.listGroupLink$.MODULE$ In listGroupLink.template.scala
		views.html.bootstrapHelper.mainContent$.MODULE$ isn't final and can't be protected from malicious code
Field isn't final and can't be protected from malicious code	Medium	In file mainContent.template.scala, lines to In class views.html.bootstrapHelper.mainContent$ Field views.html.bootstrapHelper.mainContent$.MODULE$ In mainContent.template.scala
		views.html.bootstrapHelper.sidebarPlusContent$.MODULE$ isn't final and can't be protected from malicious code
Field isn't final and can't be protected from malicious code	Medium	In file sidebarPlusContent.template.scala, lines to In class views.html.bootstrapHelper.sidebarPlusContent$ Field views.html.bootstrapHelper.sidebarPlusContent$.MODULE$ In sidebarPlusContent.template.scala
		views.html.bootstrapHelper.table$.MODULE$ isn't final and can't be protected from malicious code
Field isn't final and can't be protected from malicious code	Medium	In file table.template.scala, lines to In class views.html.bootstrapHelper.table$ Field views.html.bootstrapHelper.table$.MODULE$ In table.template.scala
		views.html.ChangePasswordAfterReset$.MODULE$ should be package protected
Field should be package protected	Medium	In file ChangePasswordAfterReset.template.scala, lines to In class views.html.ChangePasswordAfterReset$ Field views.html.ChangePasswordAfterReset$.MODULE$ In ChangePasswordAfterReset.template.scala
		views.html.CreateGroup$.MODULE$ should be package protected
Field should be package		In file CreateGroup.template.scala, lines to

protected	Medium	In class views.html.CreateGroup$ Field views.html.CreateGroup$.MODULE$ In CreateGroup.template.scala
		views.html.CreateUser$.MODULE$ should be package protected
Field should be package protected	Medium	In file CreateUser.template.scala, lines to In class views.html.CreateUser$ Field views.html.CreateUser$.MODULE$ In CreateUser.template.scala
		views.html.GroupMembersList$.MODULE$ should be package protected
Field should be package protected	Medium	In file GroupMembersList.template.scala, lines to In class views.html.GroupMembersList$ Field views.html.GroupMembersList$.MODULE$ In GroupMembersList.template.scala
		views.html.Index$.MODULE$ should be package protected
Field should be package protected	Medium	In file Index.template.scala, lines to In class views.html.Index$ Field views.html.Index$.MODULE$ In Index.template.scala
		views.html.Login$.MODULE$ should be package protected
Field should be package protected	Medium	In file Login.template.scala, lines to In class views.html.Login$ Field views.html.Login$.MODULE$ In Login.template.scala
		views.html.Main$.MODULE$ should be package protected
Field should be package protected	Medium	In file Main.template.scala, lines to In class views.html.Main$ Field views.html.Main$.MODULE$ In Main.template.scala
		views.html.OwnGroupsList$.MODULE$ should be package protected
Field should be package protected	Medium	In file OwnGroupsList.template.scala, lines to In class views.html.OwnGroupsList$ Field views.html.OwnGroupsList$.MODULE$ In OwnGroupsList.template.scala
		views.html.ResetUserPassword$.MODULE$ should be package protected
Field should be package protected	Medium	In file ResetUserPassword.template.scala, lines to In class views.html.ResetUserPassword$ Field views.html.ResetUserPassword$.MODULE$ In ResetUserPassword.template.scala
		views.html.UserCreated$.MODULE$ should be package protected
Field should be package protected	Medium	In file UserCreated.template.scala, lines to In class views.html.UserCreated$ Field views.html.UserCreated$.MODULE$ In UserCreated.template.scala
		views.html.UserList$.MODULE$ should be package protected

Field should be package protected	Medium	In file UserList.template.scala, lines to In class views.html.UserList$ Field views.html.UserList$.MODULE$ In UserList.template.scala
		views.html.UserSessions$.MODULE$ should be package protected
Field should be package protected	Medium	In file UserSessions.template.scala, lines to In class views.html.UserSessions$ Field views.html.UserSessions$.MODULE$ In UserSessions.template.scala

Performance Warnings

Warning	Priority	Details
		Should StartModule$EbeanServerProvider be a _static_ inner class?
Should be a static inner class	Medium	In file StartModule.java, lines 18 to 20 In class StartModule$EbeanServerProvider At StartModule.java:[lines 18-20]

Dodgy code Warnings

Warning	Priority	Details
		Class router.Routes$$anonfun$routes$1 has a circular dependency with other classes
Test for circular dependencies among classes	Medium	In file Routes.scala, lines 480 to 611 In class router.Routes$$anonfun$routes$1 In class router.Routes At Routes.scala:[lines 480-611]
		Class views.bootstrapHelper.Input$ has a circular dependency with other classes
Test for circular dependencies among classes	Medium	In file Input.scala, lines 6 to 20 In class views.bootstrapHelper.Input$ In class views.bootstrapHelper.Input At Input.scala:[lines 6-20]
		Dead store to dg in controllers.GroupController.deleteGroup(Long)
Dead store to local variable	Medium	In file GroupController.java, line 144 In class controllers.GroupController In method controllers.GroupController.deleteGroup(Long) Local variable named dg At GroupController.java:[line 144] At GroupController.java:[line 144]
		Dead store to currentUser in domainlogic.groupmanager.GroupManager.getGroup(Long, Long)
Dead store to local variable	Medium	In file GroupManager.java, line 77 In class domainlogic.groupmanager.GroupManager In method domainlogic.groupmanager.GroupManager.getGroup(Long, Long) Local variable named currentUser At GroupManager.java:[line 77] At GroupManager.java:[line 77]

		Dead store to users in domainlogic.groupmanager.GroupManagerTests.cannotSeeGroupMembersOfAnotherGroup()
Dead store to local variable	Medium	In file GroupManagerTests.java, line 140 In class domainlogic.groupmanager.GroupManagerTests In method domainlogic.groupmanager.GroupManagerTests.cannotSeeGroupMembersOfAnotherGroup() Local variable named users At GroupManagerTests.java:[line 140] At GroupManagerTests.java:[line 140]
		Write to static field domainlogic.loginmanager.LoginManagerTest.annika from instance method domainlogic.loginmanager.LoginManagerTest.setup()
Write to static field from instance method	Medium	In file LoginManagerTest.java, line 79 In class domainlogic.loginmanager.LoginManagerTest In method domainlogic.loginmanager.LoginManagerTest.setup() Field domainlogic.loginmanager.LoginManagerTest.annika At LoginManagerTest.java:[line 79] At LoginManagerTest.java:[line 79]
		Write to static field domainlogic.loginmanager.LoginManagerTest.lydia from instance method domainlogic.loginmanager.LoginManagerTest.setup()
Write to static field from instance method	Medium	In file LoginManagerTest.java, line 75 In class domainlogic.loginmanager.LoginManagerTest In method domainlogic.loginmanager.LoginManagerTest.setup() Field domainlogic.loginmanager.LoginManagerTest.lydia At LoginManagerTest.java:[line 75] At LoginManagerTest.java:[line 75]
		Write to static field GroupMemberPolicyTests.petersSession from instance method GroupMemberPolicyTests.setupBeforeAll()
Write to static field from instance method	Medium	In file GroupMemberPolicyTests.java, line 68 In class GroupMemberPolicyTests In method GroupMemberPolicyTests.setupBeforeAll() Field GroupMemberPolicyTests.petersSession At GroupMemberPolicyTests.java:[line 68] At GroupMemberPolicyTests.java:[line 68]
		Write to static field LoginFirewallTests.fwInstanceOne from instance method LoginFirewallTests.setup()
Write to static field from instance method	Medium	In file LoginFirewallTests.java, line 46 In class LoginFirewallTests In method LoginFirewallTests.setup() Field LoginFirewallTests.fwInstanceOne At LoginFirewallTests.java:[line 46] At LoginFirewallTests.java:[line 46]
		Write to static field LoginFirewallTests.fwInstanceTwo from instance method LoginFirewallTests.setup()
Write to static field from instance method	Medium	In file LoginFirewallTests.java, line 47 In class LoginFirewallTests In method LoginFirewallTests.setup() Field LoginFirewallTests.fwInstanceTwo At LoginFirewallTests.java:[line 47] At LoginFirewallTests.java:[line 47]
		Unread public/protected field: domainlogic.groupmanager.GroupManagerTests.mockitoRule
Unread public/protected field	Medium	In file GroupManagerTests.java, line 39 In class domainlogic.groupmanager.GroupManagerTests Field domainlogic.groupmanager.GroupManagerTests.mockitoRule At GroupManagerTests.java:[line 39]

D. Unittest Coverage Report

[all classes]

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	66.4% (73/ 110)	52.9% (364/ 688)	55.8% (1329/ 2382)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
<empty package name>	100% (5/ 5)	100% (8/ 8)	100% (62/ 62)
controllers	92.3% (12/ 13)	50.7% (37/ 73)	36.6% (89/ 243)
controllers.javascript	0% (0/ 6)	0% (0/ 38)	0% (0/ 116)
domainlogic	100% (2/ 2)	75% (3/ 4)	75% (6/ 8)
domainlogic.groupmanager	100% (2/ 2)	81.8% (9/ 11)	69.6% (78/ 112)
domainlogic.loginmanager	100% (6/ 6)	93.8% (15/ 16)	86.7% (52/ 60)
domainlogic.usermanager	100% (4/ 4)	100% (8/ 8)	98.4% (61/ 62)
extension	100% (4/ 4)	60% (9/ 15)	30% (15/ 50)
extension.test	100% (1/ 1)	100% (5/ 5)	100% (19/ 19)
models	100% (2/ 2)	83.3% (45/ 54)	77.6% (76/ 98)
models.dtos	36.4% (4/ 11)	30.5% (18/ 59)	23.6% (21/ 89)
models.finders	100% (4/ 4)	63.6% (7/ 11)	50% (11/ 22)
policy	50% (1/ 2)	94.1% (16/ 17)	94.3% (82/ 87)
policy.ext.loginFirewall	100% (4/ 4)	94.1% (16/ 17)	98.6% (69/ 70)
policy.session	75% (3/ 4)	77.8% (28/ 36)	77.1% (74/ 96)
policy.session.enforcement	100% (2/ 2)	100% (4/ 4)	91.7% (11/ 12)
router	100% (3/ 3)	83.3% (60/ 72)	77.7% (365/ 470)
views.bootstrapHelper	100% (2/ 2)	62.5% (10/ 16)	64.7% (11/ 17)
views.bootstrapHelper.enums	75% (3/ 4)	96.7% (29/ 30)	93.5% (29/ 31)
views.html	31.2% (5/ 16)	18% (18/ 100)	32.9% (141/ 428)
views.html.bootstrapHelper	30.8% (4/ 13)	20.2% (19/ 94)	24.8% (57/ 230)

generated on 2018-11-11 13:20