

Titel more titel

18. Dezember 2013

put abstract here

Inhaltsverzeichnis

1 Robust Programming	2
1.1 Model	2
1.1.1 XSS, 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
2 Sicherung mit SSL	2
2.1 SSL CA Erzeugung	2
2.2 Zertifikat erzeugung	3
2.3 Tomcat Server Config	4
3 Weitere Mögliche Massnahmen	4
3.1 Brute Force Logins	4

1 Robust Programming

1.1 Model

1.1.1 XSS, SQL-Injection und Validierung

Validierung findet statt mittels Regexp die im Java eingebaut ist. Wir benutzen sogenannte "Clean Strings". m.a.w Regexp Ausdrücke die ungültige Ch" wird defaultmässig auf "false" gesetzt in alle getter und setter methoden. Damit setzen wir voraus dass eine Validierung unbedingt stattfinden muss.

Gegen SQL Injection wird geschützt mithilfe von "Prepared Statements". Diese SQL Befehle erlauben nur die Parameter vom Benutzer im Query an bestimmten Stellen. Die Parameter werden automatisch sauber gemacht in dem alle Charakteren ein Escape als Prefix haben in den Parametern.

Schutz gegen XSS wird geleistet mithilfe von unserem Cleanstrings die auch Elemente wie Script Tags im Input verhindern.

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 böartiger 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

1.1.3 Postleitzahl Validierung

Die Postleitzahl wird wie verlangt, durch einen externen Service überprüft. 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,

2 Sicherung mit SSL

Im folgenden Abschnitte wird der Code gelistet die wir benutzt haben um ein Zertifikat zu erzeugen und dann schlussendlich Tomcat mit dem Zertifikat einzurichten.

2.1 SSL CA Erzeugung

```
OPENSSL=ca.cnf openssl req -x509 -nodes -days 3650 -newkey rsa:2048 -out apsica/certs/ca.pem  
-keyout ./apsica/private/ca.key
```

2.2 Zertifikat erzeugung

Certificate:

Data:

Version: 3 (0x2)

Serial Number: 1 (0x1)

Signature Algorithm: sha1WithRSAEncryption

Issuer: C=CH, ST=AG, L=Brugg, O=FHNW, CN=apsi.fhnw.ch/emailAddress=apsi@rolandh.tk

Validity

Not Before: Dec 18 08:10:19 2013 GMT

Not After : Dec 18 08:10:19 2014 GMT

Subject: CN=apsi.fhnw.ch, ST=AG, C=CH/emailAddress=apsi@rolandh.tk, O=FHNW

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

Public-Key: (2048 bit)

Modulus:

00:a4:43:a2:c2:93:3f:a8:52:de:f1:7b:b1:00:16:
da:c9:84:ac:b9:1e:60:08:b9:66:db:62:0b:5a:8a:
9f:17:d8:b9:49:c8:3a:2c:31:7a:ff:11:0e:aa:88:
c8:77:cf:b9:da:6e:bb:b7:94:57:82:64:c6:2f:ca:
26:b3:d8:bf:4e:0b:11:0a:cf:3a:81:a4:71:3d:47:
ff:b0:22:0f:85:4f:28:05:f4:52:0d:bb:f4:62:1f:
08:3c:3e:35:fe:10:e3:1f:13:9b:5c:07:90:a3:32:
9d:fb:00:7d:ed:7a:f0:69:ca:56:d0:b0:21:32:2b:
66:90:c3:c2:c9:0a:a2:0f:ac:34:7d:20:93:2d:fb:
73:02:78:d4:1d:b2:7e:6d:6a:89:5a:fb:09:04:94:
b2:41:7f:29:1b:09:8c:14:6a:f8:ec:c0:f7:a1:38:
6b:a4:ec:0c:fe:9d:28:6a:e6:64:a2:cc:16:11:89:
32:2e:c8:e2:52:2c:1c:6d:73:e7:32:9c:ee:32:c0:
0d:3e:4f:1b:3c:95:2f:20:2f:6f:cf:89:7f:82:74:
1d:36:0c:46:64:41:8d:98:9c:fd:15:ff:2b:83:ad:
8e:7d:0a:f3:8e:42:ac:ec:9d:a6:a4:40:16:02:84:
ce:38:da:e1:d2:f8:e7:e7:d1:b7:5a:bb:cb:90:99:
c3:1b

Exponent: 65537 (0x10001)

X509v3 extensions:

X509v3 Basic Constraints:

CA:FALSE

Signature Algorithm: sha1WithRSAEncryption

4c:c0:a7:51:25:fa:0a:61:4c:60:30:1d:3b:d8:0d:00:c7:44:
73:81:7f:2b:aa:27:65:7f:4c:82:08:6b:26:2f:a9:37:30:38:
23:58:89:17:16:48:42:45:1c:de:a1:04:11:e5:65:f0:dd:af:
e6:2a:e7:e4:cb:b1:89:e7:ef:83:c9:8c:7e:fc:42:0e:27:76:
6a:bc:db:68:af:6d:a5:78:d2:2f:e7:75:49:22:3a:91:5e:26:
69:87:ee:58:5b:9f:53:c5:5d:9d:1c:55:c7:1e:ea:af:fa:b0:
e3:6d:8c:63:d6:36:07:b4:30:4b:e0:80:83:8c:fd:cc:e7:de:
5a:3f:ef:16:35:6f:32:11:bc:0c:d2:f4:0a:d6:ee:79:05:0a:
d6:e3:36:e0:f8:68:4a:3a:ed:25:be:5f:e0:56:24:d5:1f:5e:
68:0e:9b:3c:d1:88:d3:f0:a1:54:a2:ce:5f:c6:c0:a2:79:28:
00:8e:47:cd:1e:6d:54:35:05:37:0f:cf:b2:c9:0e:96:b2:84:
69:36:1c:b0:99:39:3f:dc:1b:d8:8e:35:15:22:80:dc:71:55:
bb:34:19:da:a0:5c:84:6a:c2:e6:e5:00:8a:06:63:56:d1:93:
d6:4a:45:c5:79:3f:76:6c:59:e7:dc:9b:fd:39:69:c8:f2:f2:
86:76:aa:69

Dieser Zertifikat ist mit einem CSR verbunden und dann nachdem wir es mit unserem CA unterzeichnet haben, war es im Keystore Integriert.

2.3 Tomcat Server Config

```
<Connector port="8443" maxThreads="150" scheme="https" secure="true" SSLEnabled="true"
keystoreFile="ssl/apsi.jks" keystorePass="apsiKennwort" clientAuth="false" keyAlias="apsi"
sslProtocol="TLS"/>
```

Web.xml:

```
<security-constraint>

<web-resource-collection>

<web-resource-name>RattleBits</web-resource-name>

<url-pattern>/*</url-pattern>

</web-resource-collection>

<user-data-constraint>

<transport-guarantee>CONFIDENTIAL</transport-guarantee>

</user-data-constraint>

</security-constraint>
```

3 Weitere Mögliche Massnahmen

3.1 Brute Force Logins

Logins in Datenbank abspeichern und wenn der Anzahl ein Grenzwert erreicht hat, sollte ein boolean gesetzt werden. Dieser Benutzer muss eine Zeit dann abwarten bevor er einloggen könnte.