FHNW: Labor Applikationssicherheit

Sichere Bearbeitung von HTML-Formularen

Carlo U. Nicola 4. November 2013

Ziel

Die Bearbeitung von HTML-Formularen bildet das Rückgrat vieler Web-Applikationen. Mit dieser Laborübung sollten Sie eine erste Schutzschicht (im Sinne von Saltzer und Schröder) in Ihrer Web-Applikation sorgfältig planen und entwickeln. Sie sollen Ihre Aufmerksamkeit auf die folgenden Grundprinzipien richten:

1. *Separation of privilege*: Halten Sie Rechte sorgfältig auseinander. Konkret will heissen, dass die Programmschicht, welche die Eingaben des Benutzers sammelt, nicht auch für den eigentlichen Login des registrierten Benutzers verwendet werden darf.

2. *Complete mediation*: Jeder Input/Output muss verifiziert werden. Sie sollten im Voraus bestimmen, wie die erlaubte Information aussehen darf, und dann strikt die Kriterien für ihre Validierung formulieren.

3. *Fail-safe defaults*: Falls der Benutzer unbewusst (oder bewusst) falsche Informationen via HTML-Formular einschleusen will, sollte Ihre Applikation nur mit knappen Hinweisen reagieren und vermeiden, dass der Benutzer die Applikation allzu lange beansprucht (d.h. Sie sollten eine geeignete *fail-safe* Strategie entwerfen).

4. *Availability and reliability*: Ihre Applikation soll maximal 50 Benutzer in parallel (ohne nennenswerte Einbussen in der Performanz) bedienen können. Überlegen Sie, welche Implikationen diese Forderung für die Sicherheit Ihrer Applikation beinhaltet.

Die Abbildung 1 zeigt grob, welchen Bedrohungen eine solche Web-Applikation ausgesetzt ist.

Aufgabenstellung

1. Ihr HTML-Formular soll Kundendaten für die fiktive Firma **RattleBits** sammeln. Folgende Felder sind im Formular vorgesehen:

   a) *Firmenname*: Nur Gross- und Klein-Buchstaben und Leerzeichen (max. 20). Beispiel: Müller Martini AG;

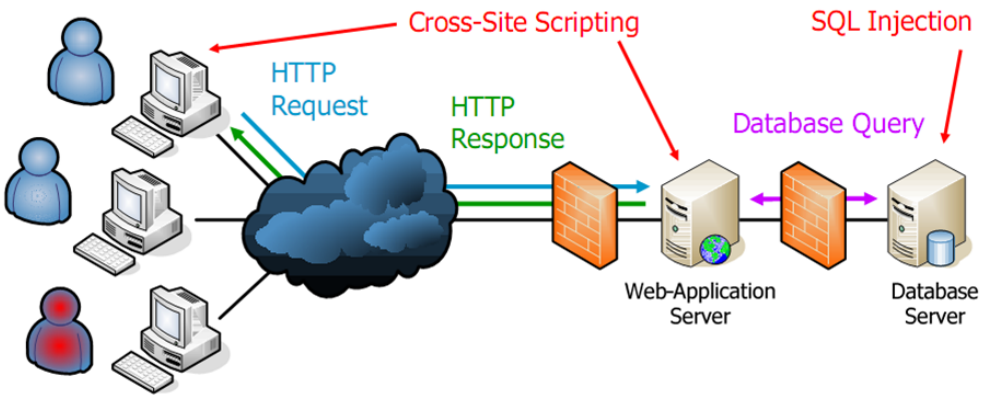

Abbildung 1: Das Bild zeigt welche Angriffspfade einem *cracker* offen stehen, um die Sicherheit einer Web-Applikation zu gefährden. Man beachte, dass Sie die Web-Applikation so robust und sicher gestalten sollten, dass Sie auf *firewalls* verzichten können!

   b) *Strasse und Strassennummer* : Erlaubte Zeichen: Gross- und Klein-Buchstaben, Zahlen, Punkt, Bindestrich, Leerzeichen;

   c) *PLZ*: Nur Zahlen, richtige PLZ für die Schweiz, nachkontrolliert mit einem Web-Dienst wie z. B.: `http://www.postleitzahlen.ch`;

   d) *Stadt*: Erlaubte Zeichen: Gross- und Klein-Buchstaben, Punkt, Bindestrich, Leerzeichen;

   e) *Email Adresse*;

2. Sie bestätigen die korrekten Angaben mit einem Email, das Sie an die (vermeintlich) real existierende Adresse senden. Sie senden aber erst, wenn Sie sicher sind, dass die Email-Adresse auch existiert (*no bouncing*). In diesem Email teilen Sie dem Benutzer seinen Spitznamen und das provisorische Passwort mit, mit denen er den *login*-Prozess einleiten kann.

3. Die Nach-Kontrolle der PLZ via Web-Dienst soll sorgfältig geplant werden. Beachten Sie folgende Punkte:

   a) Verfügbarkeit des Dienstes? Maximale Anzahl Wiederholungen; Timeout; ...

   b) Aufbau der Anfrage; Desinfektion der Antwort; ...

4. Sobald der Benutzer den (Feld)-Knopf *login* gewählt hat, wird eine HTTPS Verbindung zum Webserver hergestellt. Dann wird eine neue HTML-Form mit zwei Feldern an den Browser (via HTTPS) gesendet. Die zwei Felder sind:

a) *Benutzer Spitzname (user ID)*: min. 4 Zeichen, max. 64 Zeichen; Erlaubte Zeichen: Zahlen, Gross- und Klein-Buchstaben mit Umlauten, Punkte, Bindestriche, Unterstriche;

b) *Password*: min. 8 Zeichen, max. 64 Zeichen; Erlaubte Zeichen: Zahlen, Gross- und Klein-Buchstaben mit Umlauten, Punkten, Bindestrichen, Unterstrichen. Alles Andere ist verboten.

5. Nach dem Login für bereits registrierte Benutzer überprüfen Sie Spitzname und Passwort mit dem Eintrag in der Datenbank und falls alles i.O. ist, senden Sie ein Zitat (z. B. via `fortune` ...) Sie sollten mit der Java Klasse `PreparedStatement` (siehe: http://download.oracle.com/javase/tutorial/jdbc/basics/prepared.html) die bereits validierten Eingabe-Werte vorbereiten, bevor Sie die entsprechende SQL-Anweisung an die Datenbank senden.

6. Entwerfen Sie effiziente Massnahmen, um die Datenbank von *cracker*-Angriffen zu schützen. Hier ist es wichtig ein umfassendes Bedrohungsmodell für die Datenbank zu entwerfen. Seine vollständige Umsetzung wird nicht verlangt.

7. Erweitern Sie Ihre Applikation so, dass der Benutzer nach dem erfolgreichen *login* die Möglichkeit hat, sein Passwort sicher zu ändern.

8. Der Browser-Client und der Web-Server sollten auf zwei physisch getrennten Maschinen laufen. Dies bedeutet, dass Sie geeignete Zertifikate für den Web-Server installieren sollten, um mit `https` arbeiten zu können.

9. Sie sollen die Entwicklung des Programmes iterativ mit `FindBugs` begleiten. Die Empfehlungen von `FindBugs` betreffend Sicherheit und Robustheit sollten eingehalten werden und falls Sie sie ablehnen, dann sollten Sie die Gründe schriftlich glaubwürdig festhalten. Sie finden den `FindBugs`-Plug-in für Eclipse in http://findbugs.sourceforge.net/downloads.html.

Bedingungen

Sie sollten diese Applikation in Java[1] schreiben und zwar z. B., wie Sie bereits gelernt haben, als Servlet, das im `Tomcat` eingebettet ist. Als Datenbank muss PostgreSQL oder MySQL verwendet werden. Für die Validierung empfehle ich Ihnen das `java.util.regex.*` Paket zu benutzen, das im Unterricht besprochen worden ist.

`JavaMail` bietet viele Klassen und Methoden, die Ihnen erlauben, einfache standard Email-Meldungen zu senden (siehe http://www.oracle.com/technetwork/java/index-138643.html).

[1] Sie dürfen `JavaScript` für die Validierung nicht benutzen!

Sie sollten genau überlegen, wie Sie die provisorischen Passwörter generieren, und wie Sie die Passwort-Liste aufbauen und in System schützen wollen. Die Passwörter und Benutzernamen zusammen mit der Email-Adresse sollen auf eine kommerzielle Datenbank Ihrer Wahl (siehe oben aber keine Text Files!) abgelegt werden. Sie sollen während der Demonstration beweisen, dass Ihre Webapplikation gegen *SQL injection* und XSS-Attacken immun ist.

Weitere wichtige Merkmale der Applikation, die Sie besonders beachten sollten:

1. Umschaltung HTTP → HTTPS in Tomcat nur wenn man *login* wählt;
2. Strategie gegen mehrfache falsche login-Versuche;
3. Welche *fail/safe defaults* sollten Sie in ihrer Applikation einbauen und auch garantieren?

Was Sie abgeben müssen

1. Einen kurzen Bericht, worin Sie die wichtigsten Merkmale Ihrer Lösung festlegen (SW Aufbau, Diskussion der wichtigsten Klassen; Eventuelle Erweiterungen, usw.. Mit guten UML Diagrammen können Sie sich die Mühe des Schreibens grösstenteils ersparen ...) ;
2. Eine Demonstration, die zeigt, dass Ihre Applikation genau das tut, was Sie im Bericht auch beschrieben haben.
3. Bericht in **pdf** und **zip**-File von Quellcode per Email an `carlo.nicola@fhnw.ch` senden. Keine anderen Formate als zip und pdf werden akzeptiert!

Gruppenzusammensetzung

Zweiergruppen.

Abgabetermin

Demonstration am 16.12.2013; Bericht und Quellcode bis spätestens am 18.12.2013 24.00 MEZ.