

Security in Web Applications

Jeff Offutt

<http://www.cs.gmu.edu/~offutt/>

SWE 432

**Design and Implementation of
Software for the Web**

Topics

1. Introduction
2. User Level Security and Usability
3. Web Apps and Authentication
4. Input Data Validation
5. Exception Handling
6. Conclusions

Security Through Time

- 100 BC Rome : magic charms
- 1400s England : not much worth stealing, armed guards for the rich
- 1600s America : no doors
- 1800s USA : doors
- 1900s USA : better lock than your neighbor
- 21st Century : keys, PINs, passwords, biometrics, ...

Topics

1. Introduction
2. User Level Security and Usability
3. Web Apps and Authentication
4. Input Data Validation
5. Exception Handling
6. Conclusions

Passwords for Web Sites

- I have over 135 passwords
 - 5 bank & financial
 - 4 credit cards
 - 6 or 7 PINS
 - 4 computers
 - 9 accounts at GMU
 - 7 email
 - 14+ Other
 - 28+ Commercial
 - 5 Conferences
 - 3+ Home & Utilities
 - 27 Research & Professional
 - 11 My website
 - ...
- Most of you probably have fewer ... but still a lot
- How can we ...
 - Keep all these passwords secure ... AND
 - Remember all of them?

*That is ... balance
security and usability*

8 December 2011

© Offutt, 2011

5

Usability and Passwords

- When users are forced to change their passwords frequently, they must come up with schemes to remember
- If change is too frequent, users' schemes subvert security, making it easier to crack their passwords
- The dividing line is about six months
 - When users have to change their passwords more than twice a year, security goes down
- Designing memorable passwords is easy
- Designing secure passwords is easy

*It is **very hard** to design passwords that are both
easy to remember and secure !*

8 December 2011

© Offutt, 2011

6

User Strategies

- Have one password for all sites
- Have a simple scheme
 - april09april
 - offutt1 offutt2, ...
- Have 150 passwords and a good notebook

Frankly, none of these are very clever ...

- One password invites theft
- Simple schemes can be broken
- Nobody can remember 150 passwords ...
and notebooks get lost

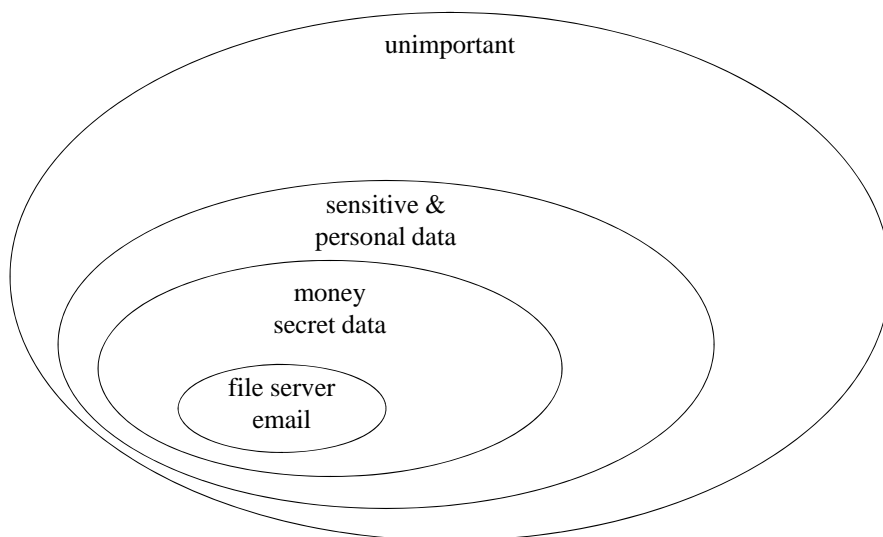
Use a multilayer approach ...

8 December 2011

© Offutt, 2011

7

Multilayer Approach to Passwords (thanks to Ravi Sandhu)



8 December 2011

© Offutt, 2011

8

Users vs. Software

- This is the user's level
 - Important principle – security always interferes with usability !
- How about the software ?
 - Important principle – security always means more work for the programmer !

Topics

1. Introduction
2. User Level Security and Usability
3. Web Apps and Authentication
4. Input Data Validation
5. Exception Handling
6. Conclusions

Security Requirements for Web Apps

1. Authentication
 - Verify the identity of the parties involved
2. Authorization
 - Limit access to resources to users
3. Confidentiality
 - Ensure that information is given only to authenticated parties
4. Integrity
 - Ensure that information is not changed or tampered with

Where to Apply?

- Security can be applied at three levels :
 1. Web server (Apache)
 2. Web container (Tomcat)
 3. Web application (your servlet)
- If implemented in a Web application, that is sometimes considered being through the container

Security Application Methods

1. Secure web applications using a Web server

- HTTP authentication
- Authorization of users/groups
- Authorization of domains
- Secure HTTP, an extension of HTTP
- SSL capabilities

2. Secure web applications using a servlet container

- HTTP authentication (basic, digest)
- Form-based authentication
- Authorization of users/groups
- SSL capabilities

3. Securing web applications by programming

- Authorization of users
- User information kept on the server in a session

8 December 2011

© Offutt, 2011

13

User-level HTTP Passwords: Apache (1. web server authentication)

- In the directory : create `.htaccess` :
AuthUserFile /home/student/gburdell/lib/users — passwd file, readable
AuthGroupFile /dev/null — for groups, forget it
AuthName swe642 — name of directory, part of prompt
AuthType Basic — the only one allowed
<Limit GET> — most common access control
require user gburdell
</Limit>
• Create password :
`htpasswd -c /home/student/gburdell/lib/users gburdell`
Will prompt for the password
• Adding additional users :
 1. `htpasswd /home/student/gburdell/lib/users george`
 2. add to `.htaccess`: `require user george`

8 December 2011

© Offutt, 2011

14

Authentication in Web Apps by Programming

- Use the password input field
 - `<input name="password" type="password" id="password">`
- Validate the username and password on the server
- Store whether the user has been authenticated in the session object
 - **Never** pass this information back to the client !
- Don't forget to lock the *back doors*
 - Check authentication in every software component
 - If the user is not authenticated, go back to the login screen
 - This is the most common vulnerability in web applications

8 December 2011

© Offutt, 2011

15

Secure Socket Layer (SSL) based Authentication (*https*)

- Invented by Netscape in the mid 90's
- Encrypt every HTTP message to and from the web server using standard PKI technology
- De-facto standard used for secure web-based transactions
- Default URL `https://some.domain.com` with default port number 443
- Don't get confused with S-HTTP
 - Encrypts only the http message body

8 December 2011

© Offutt, 2011

16

Applicability of Client SSL Authentication

- Highest level of security
- Can integrate with smart-card and biometric technologies
- Now supported by most browsers
 - Plug-ins are sometimes required
- Additional server module required to validate clients, usually using a vendor-specific security server
- Very expensive because large PKI resources (hardware / software / personnel) needed to create, maintain, distribute user certificates

8 December 2011

© Offutt, 2011

17

Security in Web Applications

- Web applications require proper security at various levels for different purposes
 - HTTP Authentication, Basic/Digest (lowest level)
 - Form-based authentication
 - Customized authentication
 - SSL server authentication
 - SSL client authentication (highest level)
- Other security concerns
 - Database security
 - Network security
 - Human factors
 - Users must remember passwords
 - Changing passwords more than twice a year decreases security
- More sophisticated techniques are available

8 December 2011

© Offutt, 2011

18

Topics

1. Introduction
2. User Level Security and Usability
3. Web Apps and Authentication
4. Input Data Validation
5. Exception Handling
6. Conclusions

Validating Inputs

Input Validation

Deciding if input values can be processed by the software

- Before starting to process inputs, wisely written programs check that the inputs are valid
- How should a program recognize invalid inputs ?
- What should a program do with invalid inputs ?
- It is easy to write input validators – but also easy to make mistakes !

Representing Input Domains

- Goal domains are often irregular
- Goal domain for credit cards[†]
 - First digit is the Major Industry Identifier
 - First 6 digits and length specify the issuer
 - Final digit is a “check digit”
 - Other digits identify a specific account
- Common specified domain
 - First digit is in { 3, 4, 5, 6 } (travel and banking)
 - Length is between 13 and 16
- Common implemented domain
 - **All digits are numeric**

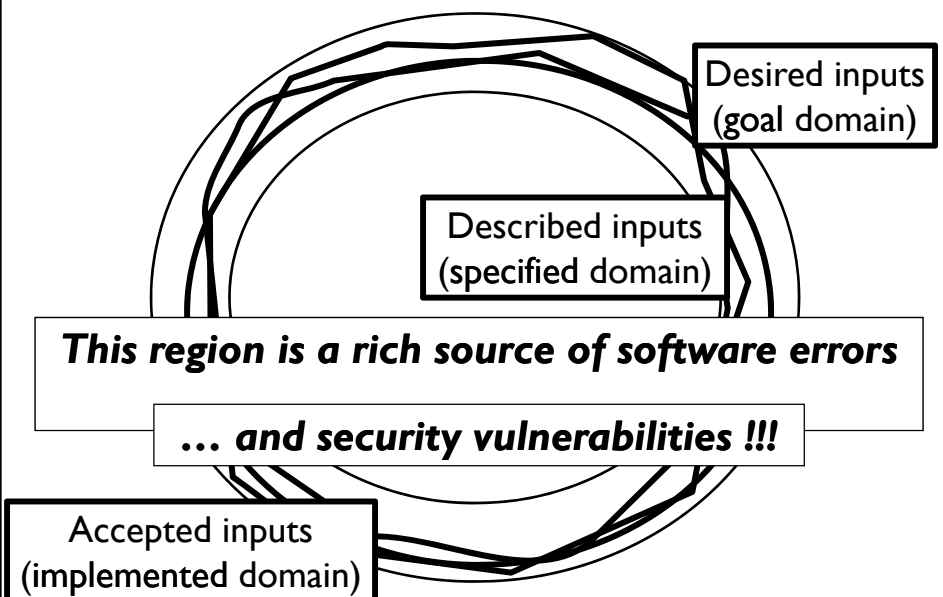
[†] More details are on : <http://www.merriampark.com/anatomycc.htm>

8 December 2011

© Offutt, 2011

21

Representing Input Domains



8 December 2011

© Offutt, 2011

22

Users can Bypass Client Validation

- Client-side HTML and Javascript can impose constraint enforcement
 - JS checks on input values
 - HTML restrictions such as maxLength
 - Implicit restrictions such as dropdown menus and radio boxes
- Users can violate constraints (accidentally and intentionally):
 - When automating
 - Turning JS off
 - To attack your software

8 December 2011

© Offutt, 2011

23

Example

User Name: Age:

Version to purchase:

Small	Medium	Large
\$150	\$250	\$500
<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>

8 December 2011

© Offutt, 2011

24

Client Side Checking

Invalid data, please correct ...

User Name:

Age:

Username should be plain text only.

Age should be between 18 and 150.

Version to purchase:

Small

Medium

Large

\$150

\$250

\$500



8 December 2011

© Offutt, 2011

25

Abbreviated HTML

```
<FORM >
```

```
<INPUT Type="text" Name="username" Size=20>
```

```
<INPUT Type="text" Name="age" Size=3 Maxlength=3>
```

```
<P> Version to purchase:
```

```
...
```

```
<INPUT Type="radio" Name="version" Value="150" Checked>
```

```
<INPUT Type="radio" Name="version" Value="250">
```

```
<INPUT Type="radio" Name="version" Value="500">
```

```
<INPUT Type="submit" onClick="return checkInfo(this.form)">
```

```
<INPUT Type="hidden" isLoggedIn="no">
```

```
</FORM>
```

Constraints

8 December 2011

© Offutt, 2011

26

Saved & Modified HTML

```
<FORM >
  <INPUT Type="text" Name="username" Size=20>
  <INPUT Type="text" Name="age" Size=3 Maxlength=3>
  <P> Version to purchase:
  ...
  <INPUT Type="radio" Name="version" value=25 Checked>
  <INPUT Type="submit" onClick="return checkInfo (this.form)">
  <INPUT Type="hidden" isLoggedIn= yes >
</FORM>
```

Allows an input with arbitrary age, no checking, cost=\$25 ...
'<' can crash an XML parser
Text fields can have SQL statements

8 December 2011

© Offutt, 2011

27

SQL Injection

User Name: Password:

Original SQL:

```
SELECT username FROM adminuser WHERE
username='turing' AND password ='enigma'
```

"injected" SQL:

```
SELECT username FROM adminuser WHERE username='turing'
OR '1' = '1' AND password ='enigma' OR '1' = '1'
```

8 December 2011

© Offutt, 2011

28

Do not trust users !!!

Apply input validation to
all inputs

Topics

1. Introduction
2. User Level Security and Usability
3. Web Apps and Authentication
4. Input Data Validation
5. Exception Handling
6. Conclusions

Managing Exceptions

- Language exception handling features allow programmers to separate functional logic from error condition handling

```
try
{
    A computation that can produce exception
}
catch (BadException e)
{
    log it and recover
}
```

- Java compiler verifies exceptions handled in program
- Some languages do not support this
- Checked exceptions force engineers to handle errors

8 December 2011

© Offutt, 2011

31

Catch Low—If You Can Recover

- Have a sensible recovery strategy
 - FileNotFoundException : Ask user for another file name
 - System.OutOfMemoryException : Probably kill the process
- Catching “low” means you have more information to recover with
 - But do not catch just to catch
 - If you don’t know what to do with the exception, let somebody else take it
- What does the user need to know ?
- Make sure you catch all exceptions at the top level

8 December 2011

© Offutt, 2011

32

Hide Exception Data From Users

Application: photosprintshopWeb

Error: java.lang.IllegalStateException exception

Reason:

java.lang.IllegalStateException: An Exception occurred while generating the Exception page 'WOExceptionPage'. This is most likely due to an error in WOExceptionPage itself. Below are the logs of first the Exception in WOExceptionPage, second the Exception in Application that triggered everything.

com.webobjects.foundation.NGForwardException

[com.webobjects.jdbcadaptor.JDBCAdaptorException] dateInformation of type java.lang.String is not a valid Date type. You must use java.sql.Timestamp, java.sql.Date, or java.sql.Time: <Session> failed instantiation. Exception raised : com.webobjects.jdbcadaptor.JDBCAdaptorException: dateInformation of type java.lang.String is not a valid Date type. You must use java.sql.Timestamp, java.sql.Date, or java.sql.Time: com.webobjects.jdbcadaptor.JDBCAdaptorException: dateInformation of type java.lang.String is not a valid Date type. You must use java.sql.Timestamp, java.sql.Date, or java.sql.Time

Original Exception:

com.webobjects.jdbcadaptor.JDBCAdaptorException: dateInformation of type java.lang.String is not a valid Date type. You must use java.sql.Timestamp, java.sql.Date, or java.sql.Time

8 December 2011

© Offutt, 2011

33

List Thrown Exceptions Explicitly

- Lazy approach :
 - throws Exception
- Engineering approach :
 - throws IOException, SQLException, IllegalAccessException
- This is about communication
 - The caller (clients) must know what they need to catch
- Be careful with finally
 - Returning from a finally block means NO exceptions will propagate to the parent

8 December 2011

© Offutt, 2011

34

Always Log Exceptions

- They usually indicate an error in the program or an error by the user
 - “Errors” by users could be attacks
 - Errors by users could highlight usability flaws
- An exception can be made if the exception handling is part of normal processing
 - Some teachers encourage this, some discourage it

Topics

1. Introduction
2. User Level Security and Usability
3. Web Apps and Authentication
4. Input Data Validation
5. Exception Handling
6. Conclusions

Security Over the Years

- Web applications open up many avenues for security threats
- In the 1980s, security was all math ...
- In the 1990s, security revolved around the database ...
- In the 2000s, security moved to the network ...
- Now most security vulnerabilities are due to software faults
- For more information, take SWE 781, Secure Software Design and Programming

Summary

In 2007, Symantec reported that most security vulnerabilities were now due to software faults

In a house : your lock should be better than your neighbor's
This principle does not work with Web apps