Security for Web-based Applications

Jeff Offutt & Nick Duan

http://www.ise.gmu.edu/~offutt/

SWE 432
Design and Implementation of Software for the Web

Security Through Time

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

Passwords for Web Applications

- I have over 80 passwords
  - > 10 for conferences I review for
  - 5 for parts of my Web site
  - >15 research related (journals, NSF, etc.)
  - > 15 commercial & email
  - 5 Web applications at GMU
  - 4 or 5 PCs
  - ise, ite, osf1, apps cluster
  - > 5 accounts that I have access to
- 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

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 !*

Strategies

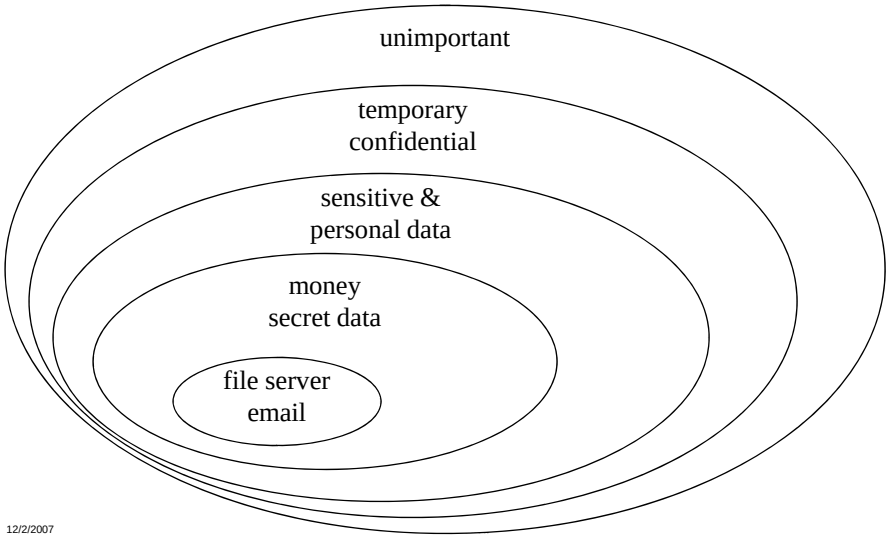
- Have one password for all sites
- Have a simple scheme
  - april08april, may08may
  - offutt1 offutt2, …
- Have 50 passwords and a good notebook

Frankly, none of these are very clever …

- One password invites theft
- Simple schemes can be broken
- Nobody can remember 50 passwords … and notebooks get lost

Use a multilayer approach …

Multilayer Approach to Passwords



User vs. Software

- This is the user's level
- How about the software ?

Security Requirements for Web Apps

- Authentication
  - Verify the identity of the user
- Authorization
  - Allow access to resources only to allowed users
- Confidentiality
  - Ensure that information is passed only to allowed users
- Integrity
  - Ensure that data is not inappropriately changed

Where to Apply?

- Security can be applied at three levels :
  - Web server (Apache)
  - Web container (Tomcat)
  - Web application (your servlet)
- Each level offers different amounts of security and flexibility

Security Application Methods

- Secure web applications using a web server
  - HTTP authentication
  - Authorization of users/groups
  - Authorization of domains
  - Secure HTTP, an extension of HTTP
  - SSL capabilities
- Secure web applications using a servlet container
  - HTTP authentication (basic, digest)
  - Form-based authentication
  - Authorization of users/groups
  - SSL capabilities
- Securing web applications by programming
  - Authorization of users
  - User information kept on the server in a session

Web Server
User-level HTTP Passwords with Apache

- In the directory : create **.**htaccess :

```
AuthUserFile /home/student/gpb/lib/users - passwd file, encrypted, readable
AuthGroupFile /dev/null      - for groups, forget it
AuthName swe432              - name of directory, part of prompt
AuthType Basic               - the only one allowed
<Limit GET>                  - most common access control
  require user gburdell
</Limit>
```

- Create password from the command line :

```
/usr/local/apache2/bin/htpasswd -c users gburdell
```

  users : passwd file
  gburdell : user name
  Will prompt for the password

User-level HTTP Passwords with Apache (2)

- Adding new users :

```
/usr/local/apache2/bin/htpasswd users george
```

  add to **.**htaccess : require user george

- This is how the class website password works

- This is specific to the web server
  - tomcat for our class website

Securing Web Applications 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

Secure Socket Layer (SSL) based Authentication

- 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

Applicability of Client SSL Authentication

- Highest level of security
- Possibility to integrate with smart-card and biometric technologies
- Now supported by most browsers
  - Through plug-ins 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

Summary

- Web applications requires proper security at various levels for different purposes
  - HTTP Authentication (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 have to remember passwords
    - Changing passwords more than twice a year *decreases* security
- Most security vulnerabilities are due to software faults

Be safe … Be secure!

In a house : your lock should be better than your neighbor’s

This principle does not work with Web apps