Web Login, Cookies

COMP427/541
2015. 01. 26.
Web Login | Old way

<form method="POST" action="postlogin.php">
    <input type="text" name="username">
    <input type="password" name="password">
</form>

http://resources.infosecinstitute.com/vulnerable-encoded-url/
http://blog.parhammajd.co.uk/css/a-simple-login-form-css/
What’s Wrong

• User ID and password is transferred in plaintext
  – Anybody can learn your password

• Adversary can directly send HTTP request with password in URL
  – Easy target for brute force attack

• Very common in 90s to early 2000s

https://samsclass.info/123/proj10/p3-sniff.htm
Simple Fixes

• Secure Hash
  – Send hash of the password instead of plaintext
  – But...

• HTTPS
  – Associate SSL/TLS session before login
    • Most of the web site performs this
  – But...
TLS One-Way Authentication

• TLS requires digital signature for authentication
• To sign, signing certificate is required
  – Issuing user certificates is expensive
• Current TLS(HTTPS) authentication
  – Only server signs, client does not
    → Client authenticates server, but server does not

• Threat
  – Nasty proxy server can re-encrypt, re-sign all the packets
    • Most of the hotels and IT kiosks does this thing
    • And many big companies too
Simple Attack | SQL Injection

```java
String txtUserId = getRequestString("UserId");
String txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

SQL Injection

Hi, this is your son's school. We're having some computer trouble.

Oh, dear — did he break something? In a way—

Did you really name your son Robert?; DROP TABLE Students; --?

Oh, yes, little Bobby tables, we call him.

Well, we've lost this year's student records. I hope you're happy.

And I hope you've learned to sanitize your database inputs.

https://xkcd.com/327/

http://www.abluestar.com/blog/sql-injection-license-plate/
SQL Injection Protection

• Simple solution: Blacklist
  – Attacker can eventually circumvent

• Input Sanitization
  – Modern database engine supports input sanitization feature
    • i.e) @ marker in JavaScript
    • i.e) %Q format string in SQLite
How CSRF Works

GET / HTTP/1.1
Host: www.evil.org

GET/transfer?to=hacker&amount=1000$ HTTP/1.1
Host: bank.com

HTTP/1.1 200 OK
...
<html>
...
<img src="http://bank.com/transfer?to=hacker&amount=1000$"/>
...
</html>

http://www.slideshare.net/BjrnKimminich/web-application-security-21684264
Protection From CSRF

• Add a small token for each request
  – Should not be automatic
  – Should be cryptographically strong
  – Should not be exposed easily

• CSRF with XSS will be very powerful

http://www.slideshare.net/BjrnKimminich/web-application-security-21684264
XSS Flow Example

http://www.slideshare.net/BjrnKimminich/web-application-security-21684264
XSS Pattern

• Simple Patterns
  – `<SCRIPT>javascript:alert('XSS');</SCRIPT>`
  – `<IMG SRC=javascript:alert('XSS')>`
  – `<IFRAME SRC="javascript:alert('XSS');"></IFRAME>`

• Masked / Evasive Patterns
  – `<IMG SRC=javascript:alert(‘XSS’);>`
  – `<IMG """>alert("XSS")</SCRIPT>"`
  – `<IMG SRC="javascript:alert(‘XSS’);">`
  – `<IMG SRC="jav\x09ascript:alert(‘XSS’);">`

• Whitelisting of the input is required
  – There are sanitizer tools (i.e) OWASP HTML JAVA sanitizer

http://www.slideshare.net/BjrnKimminich/web-application-security-21684264
HTTP Session Management

• HTTP is “stateless” protocol
  – Cannot keep information over a series of accesses
• Server should maintain “session” somehow

• Simple old solution
  – Use hidden form in HTML
    • Web server assigns random value on this form
    • User does not see the value
    • BUT...
HTTP Cookie

- A special value created by server and sent to client
- Client stores the value and attaches it on later access to the same domain

[Diagram]

1. The browser requests a web page
2. The server sends the page and the cookie
3. The browser requests another page from the same server

http://www.slideshare.net/RitikaBarethia/presentation-on-internet-cookies
HTTP Cookie | scope

- Expiry information
  - (YYYY/MM/DD/, HH/MM/SS)
  - Usually 20 minutes

- Path Information
  - Available path in the domain (i.e /customer/php/)

- Domain Information
  - i.e) comp427.edu

- Etc.
What’s Wrong This Time

• It’s still plaintext
  – Attacker can easily impersonate / hijack
  – HTTPS can protect this (but not perfectly)

https://pineappleorange.wordpress.com/tag/wireshark/
Firesheep (2010)

• A Firefox extension by Eric Butler
  – Eavesdrop network traffic
  – Extract unencrypted cookie
  – Show up hijacked user credential
  – Attacker can access the site

http://www.pcworld.com/article/208773/Firesheeps_a_Huge_Hit_with_Amateur_Hackers.html
Solutions

• Bearer token
  – Server creates random token to client, other than cookie
  – Risky when attacker learns the token design

• Existing solutions
  – i.e.) Kerberos: Old, but secure authentication protocol

• Proprietary solutions
  – Powered by HTML5, many major web sites provides their own solutions
Kerberos (v5, 1993)

• Authentication protocol  
  – Part of MIT Project (Athena)

• Secure Session Management  
  – User needs to enter password only 1 time

• Widely used by many systems

• Does not require public key cryptography and hash functions  
  – Based on 80’s technology
Kerberos Protocol

1. User logs on to workstation and requests service on host.
2. AS verifies user’s access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user’s password.
3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user’s name, network address, and time to TGS.
4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.
5. Workstation sends ticket and authenticator to server.
6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

https://courses.cs.washington.edu/courses/csep590/06wi/lectures/slides/LaMacchia_012406.ppt
Password Management

• Users manage ID and password for each web site
  – Mostly same (easy) password and never change
    • Each web site’s password policy is different
  – Infeasible to remember all the passwords if a user manages them differently
  – Password recovery questions are very stupid
    • i.e) “Name of the city you born” for Monaco citizen

• Password management application is required
  – Do you trust your password manager?
Trend Micro Password Manager

• Google found vulnerability in Trend Micro Password Manager (01.2016)
  – Password manager itself is a privileged start up service
  – It is basically a web engine
  – Accepts JavaScript from the web sites!!!

```javascript
x = new XMLHttpRequest()
x.open("GET",
"https://localhost:49155/api/openUrlInDefaultBrowser?url=c:\windows\system32\calc.exe true);
try { x.send(); } catch (e) {};
```

https://code.google.com/p/google-security-research/issues/detail?id=693
OpenID

- Decentralized single sign on (SSO) mechanism
  - User ID is a URI
    - i.e) http://danwallach.comp427.edu
    - Any URI is possible
  - Web server asks the URI for authenticity
    - User information does not go to web sites
    - There are extension APIs to transfer user information
      - Under user consent

- Open standard
  - 100+ Standard APIs
  - Nobody owns the mechanism
  - You can create your own OpenAPI server
OpenID Architecture

http://designpatternschaos.blogspot.com/2009_04_01_archive.html
OAuth

• Simple open standard for secure authentication
  – Log in part of website
    • i.e) photos, mails, bookmarks, locations, ...
  – Token based authentication
    • Does not send ID and password to the web site
  – Open source, open API
Problems

• NASCAR Problem
  – User is overwhelmed by so many choices

• Usability
  – URI is not good to remember by human
2(n) Factor Authentication

- 3 different types of the password
  - Something you have
  - Something you know
  - Something you are

- 2 factor authentication is combining 2 types of password for authentication
  - Password + SMS token
  - Password + HW token
  - Password + fingerprint
  - ...

http://www.theverge.com/2014/10/21/7027267/google-launches-support-for-security-key-a-simpler-kind-of-two-factor
Question?