Web Security:
Sessions; CSRF;
start on authentication

CS 161: Computer Security
Prof. Raluca Ada Popa

Nov 10, 2016

Credit: some slides are adapted from previous offerings of this course or from CS 241 of Prof. Dan Boneh
Announcements

Proj 3 due on Thur, Nov 17
Recall: Cookie scope

GET ...

HTTP Header:
Set-cookie: NAME=VALUE ;
domain = (when to send) ;
path = (when to send)
secure = (only send over SSL);
expires = (when expires) ;
HttpOnly

- Expires is expiration date
  - Delete cookie by setting “expires” to date in past
- HttpOnly: cookie cannot be accessed by Javascript, but only sent by browser
Recall: What scope a server may set for a cookie

domain: any domain-suffix of URL-hostname, except TLD
[div>top-level domains, e.g. `.com`]

example: host = "login.site.com"

<table>
<thead>
<tr>
<th>allowed domains</th>
<th>disallowed domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>login.site.com</td>
<td>user.site.com</td>
</tr>
<tr>
<td>.site.com</td>
<td>othersite.com</td>
</tr>
<tr>
<td></td>
<td>.com</td>
</tr>
</tbody>
</table>

path: can be set to anything
Recall: When browser sends cookie

A cookie with
  domain = example.com, and
  path = /some/path/
will be included on a request to
  http://foo.example.com/some/path/subdirectory/hello.txt
Client side read/write: document.cookie

- Setting a cookie in Javascript:
  ```javascript
document.cookie = "name=value; expires=...;"
  ```

- Reading a cookie:
  ```javascript
alert(document.cookie)
  ```
  prints string containing all cookies available for document (based on [protocol], domain, path)

- Deleting a cookie:
  ```javascript
document.cookie = "name=; expires= Thu, 01-Jan-00"
  ```

document.cookie often used to customize page in Javascript
Viewing/deleting cookies in Browser UI

Firefox: Tools -> page info -> security -> view cookies
Sessions
Sessions

- A sequence of requests and responses from one browser to one (or more) sites
  - Session can be long (Gmail - two weeks) or short (e.g., banking)
  - without session mgmt:
    - users would have to constantly re-authenticate

- Session management:
  - Authorize user once;
  - All subsequent requests are tied to user
Pre-history: HTTP auth

HTTP request: GET /index.html

HTTP response contains:

WWW-Authenticate: Basic realm="Password Required"

Browsers sends hashed password on all subsequent HTTP requests:

Authorization: Basic ZGFddfibzsdfgkjheczi1NXRleHQ=

What problems can you see with this model?
HTTP auth problems

- Hardly used in commercial sites
  - User cannot log out other than by closing browser
    - What if user has multiple accounts?
    - What if multiple users on same computer?
  - Site cannot customize password dialog
  - Confusing dialog to users
  - Easily spoofed
Session tokens

**Browser**
- GET /index.html
- GET /books.html
- POST /do-login
  - Username & password
  - elevate to a logged-in session token
- POST /checkout
  - logged-in session token

**Web Site**
- set anonymous session token
- anonymous session token
- check credentials
- Validate token
Storing session tokens:  
Lots of options  (but none are perfect)

• Browser cookie:  
  Set-Cookie: SessionToken=fduhe63sfdb

• Embedd in all URL links:  
  https://site.com/checkout ? SessionToken=kh7y3b

• In a hidden form field:  
  <input type="hidden" name="sessionid" value="kh7y3b">

Can you see problems with these?
Storing session tokens: problems

- **Browser cookie:**
  browser sends cookie with every request, even when it should not (see CSRF attack)

- **Embed in all URL links:**
  token leaks via HTTP Referer header (your browser tells a site which previous site it visited last in the Referer header, which may contain session tokens)

- **In a hidden form field:** short sessions only

Best answer: a combination of all of the above.
Cross Site Request Forgery
**Top web vulnerabilities**

<table>
<thead>
<tr>
<th>OWASP Top 10 – 2010 (Previous)</th>
<th>OWASP Top 10 – 2013 (New)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1 – Injection</td>
<td>A1 – Injection</td>
</tr>
<tr>
<td>A3 – Broken Authentication and Session Management</td>
<td>A2 – Broken Authentication and Session Management</td>
</tr>
<tr>
<td>A2 – Cross-Site Scripting (XSS)</td>
<td>A3 – Cross-Site Scripting (XSS)</td>
</tr>
<tr>
<td>A4 – Insecure Direct Object References</td>
<td>A4 – Insecure Direct Object References</td>
</tr>
<tr>
<td>A6 – Security Misconfiguration</td>
<td>A5 – Security Misconfiguration</td>
</tr>
<tr>
<td>A7 – Insecure Cryptographic Storage – Merged with A9</td>
<td>A6 – Sensitive Data Exposure</td>
</tr>
<tr>
<td>A8 – Failure to Restrict URL Access – Broadened into</td>
<td>A7 – Missing Function Level Access Control</td>
</tr>
<tr>
<td><strong>A5 – Cross-Site Request Forgery (CSRF)</strong></td>
<td><strong>A8 – Cross-Site Request Forgery (CSRF)</strong></td>
</tr>
<tr>
<td>&lt;buried in A6: Security Misconfiguration&gt;</td>
<td>A9 – Using Known Vulnerable Components</td>
</tr>
</tbody>
</table>
HTML Forms

Allow a user to provide some data which gets sent with an HTTP POST request to a server

```html
<form action="bank.com/action.php">
    First name:  <input type="text" name="firstname">

    Last name:<input type="text" name="lastname">

    <input type="submit" value="Submit"></form>
```

When filling in Alice and Smith, and clicking submit, the browser issues

HTTP POST request bank.com/action.php?firstname=Alice&lastname=Smith

As always, the browser attaches relevant cookies
Recall: session using cookies

Browser

POST/login.cgi

Set-cookie: authenticator

GET...
Cookie: authenticator

response

Server
Basic picture

1. establish session
2. visit server
3. receive malicious page containing URL to bank.com with bad actions
4. send forged request (w/ cookie)

What can go bad? URL contains transaction action, bank checks cookie
Cross Site Request Forgery (CSRF)

Example:

- User logs in to bank.com
  - Session cookie remains in browser state

- User visits malicious site containing:
  
  ```html
  <form name=F action=http://bank.com/BillPay.php>
  <input name=recipient value=badguy> ...
  <script> document.F.submit(); </script>
  ```

- Browser sends user auth cookie with request
  - Transaction will be fulfilled

Problem:

- cookie auth is insufficient when side effects occur
Form post with cookie

GET /blog HTTP/1.1
Form post with cookie

www.attacker.com → Victim Browser → www.bank.com

GET /blog HTTP/1.1

<form action=https://www.bank.com/transfer method=POST target=invisibleframe>
  <input name=recipient value=attacker>
  <input name=amount value=$100>
</form>

<script>document.forms[0].submit();</script>

POST /transfer HTTP/1.1
Referer: http://www.attacker.com/blog
Recipent=attacker&amount=$100
Cookie: SessionID=523FA4cd2E

HTTP/1.1 200 OK
Transfer complete!

User credentials
An attacker could
• add videos to a user’s "Favorites,"
• add himself to a user’s "Friend" or "Family" list,
• send arbitrary messages on the user’s behalf,
• flagged videos as inappropriate,
• automatically shared a video with a user’s contacts,
  subscribed a user to a "channel" (a set of videos
  published by one person or group), and
• added videos to a user’s "QuickList" (a list of videos
  a user intends to watch at a later point).
Facebook Hit by Cross-Site Request Forgery Attack

By Sean Michael Kerner  I  August 20, 2009

September 30, 2008

Popular websites fall victim to CSRF exploits
Defenses
CSRF Defenses

- Secret Validation Token
  - `<input type=hidden value=23a3af01b>`

- Referer Validation

- Others (e.g., custom HTTP Header)
  - `X-Requested-By: XMLHttpRequest`
Secret Token Validation

The server requests a secret token for every action, the user’s browser obtained this token if the user visited the site and browsed to that action, instead of directly sending an action; attacker won’t have the token

1. goodsite.com server includes a secret token into the webpage (e.g., in forms as a hidden field)
2. Requests to goodsite.com include the secret
3. goodsite.com server checks that the token embedded in the webpage is the expected one; reject request if not

Can the token be?

- 123456
- Dateofbirth

Validation token must be hard to guess by the attacker
How token is used

- The server stores state that binds the user's CSRF token to the user's session id
- Embeds CSRF token in every form
- On every request the server validates that the supplied CSRF token is associated with the user's session id
- Disadvantage is that the server needs to maintain a large state table to validate the tokens.
When the browser issues an HTTP request, it includes a referer header that indicates which URL initiated the request.

This information in the Referer header could be used to distinguish between same site request and cross site request.
Referer Validation

Facebook Login

For your security, never enter your Facebook password on sites not located on Facebook.com.

Email: 
Password: 

[ ] Remember me

Login or Sign up for Facebook

Forgot your password?
Referer Validation Defense

HTTP Referer header

- Referer: http://www.facebook.com/
- Referer: http://www.attacker.com/evil.html
- Referer:
  - Strict policy disallows (secure, less usable)
  - Lenient policy allows (less secure, more usable)
Privacy Issues with Referer header

- The referer contains sensitive information that impinges on the privacy
- The referer header reveals contents of the search query that lead to visit a website.
- Some organizations are concerned that confidential information about their corporate intranet might leak to external websites via Referer header
Referer Privacy Problems

- Referer may leak privacy-sensitive information
  

- Common sources of blocking:
  - Network stripping by the organization
  - Network stripping by local machine
  - Stripped by browser for HTTPS -> HTTP transitions
  - User preference in browser

Hence, such block might help attackers in the lenient policy case
Custom HTTP Headers

- Browsers prevent sites from sending custom HTTP headers to another site but allow sites to send custom HTTP headers to themselves.
- Cookie value is not actually required to prevent CSRF attacks, the mere presence of the header is sufficient.
- To use this scheme as a CSRF Defense, a site must issue all state modifying requests using XMLHttpRequest, attach the header and reject all requests that do not accompany the header.
Custom Header Defense

- XMLHttpRequest is for same-origin requests
  - Can use setRequestHeader within origin
- Limitations on data export format
  - No setRequestHeader equivalent
  - XHR2 has a whitelist for cross-site requests
- Issue POST requests via AJAX:

- Doesn't work across domains

X-Requested-By: XMLHttpRequest
Summary: sessions and CSRF

- Cookies add state to HTTP
  - Cookies are used for session management
  - They are attached by the browser automatically to HTTP requests
- CSRF attacks execute request on benign site because cookie is sent automatically
- Defenses for CSRF:
  - embed unpredicatable token and check it later
  - check referer header
Authentication & Impersonation
Authentication

- Verifying someone really is who they say they claim they are
- Web server should authenticate client
- Client should authenticate web server
Impersonation

- Pretending to be someone else
- Attacker can try to:
  - Impersonate client
  - Impersonate server
Authenticating users

How can a computer authenticate the user?

- “Something you know”
  - e.g., password, PIN
- “Something you have”
  - e.g., smartphone, ATM card, car key
- “Something you are”
  - e.g., fingerprint, iris scan, facial recognition
Two-factor authentication

Authentication using two of:

- Something you know (account details or passwords)
- Something you have (tokens or mobile phones)
- Something you are (biometrics)
Example

Online banking:
- Hardware token or card ("smth you have")
- Password ("smth you know")

Mobile phone two-factor authentication:
- Password ("smth you know")
- Code received via SMS ("smth you have")
Is this a good 2FA?

- Password
- Answer to security question

This is not two-factor authentication because both of the factors are something you know.
After authenticating...

- Session established
  - Session ID stored in cookie
  - Web server maintains list of active sessions (sessionID mapped to user info)
- Reauthentication happens on every http request automatically
  - Recall that every http request contains cookie
After authenticating...

Alice

sessionID = 3458904043
Must be unpredictable

Session hijacking attack:
- Attacker steals sessionID, e.g., using a packet sniffer
- Impersonates user

Active sessions:
<table>
<thead>
<tr>
<th>sessionID</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>3458904043</td>
<td>Alice</td>
</tr>
<tr>
<td>5465246234</td>
<td>Bob</td>
</tr>
</tbody>
</table>
After authenticating...

Active sessions:
- 3458904043 | Alice
- 5465246234 | Bob

**sessionID = 3458904043**
Must be unpredictable

Protect sessionID from packet sniffers:
- Send encrypted over HTTPS
- Use *secure* flag to ensure this

When should session/cookie expire?
- Often is more secure
- But less usable for user

Other flags?
- `httponly` to prevent scripts from getting to it
After authenticating...

Alice

sessionID = 3458904043
Must be unpredictable

Active sessions:
3458904043 | Alice
5465246234 | Bob

What if attacker obtains old sessionID somehow?

• When user logs out, server must remove Alice’s entry from active sessions
• Server must not reuse the same session ID in the future
• Old sessionID will not be useful
Authenticating the server

Why should user authenticate the web server she is interacting with?

- User is introducing sensitive data to server including credentials for performing actions
Phishing

- Attacker creates fake website that appears similar to a real one
- Tricks user to visit site (e.g. sending email)
- User inserts credentials and sensitive data which gets sent to attacker
- Web page then directs to real site or shows maintenance issues
http://ebay.attacker.com/
Welcome to eBay

Ready to bid and buy? Register here

Join the millions of people who are already a part of the eBay family. Don't worry, we have room for one more.

Register as an eBay Member and enjoy privileges including:

- Bid, buy and find bargains from all over the world
- Shop with confidence with PayPal Buyer Protection
- Connect with the eBay community and more!

Sign in to your account

Back for more fun? Sign in now to buy, bid and sell, or to manage your account.

User ID

I forgot my user ID

Password

I forgot my password

Keep me signed in for today. Don't check this box if you're at a public or shared computer.

Having problems with signing in? Get help.

Protect your account: Create a unique password by using a combination of letters and numbers that are not
Please confirm your identity jbieber

Please answer security question below.

What is your mother’s maiden name?  

Smith

Answer the secret question you provided.

What is your other eBay user ID or another's member in your household?  

NA

What email used to be associated with this account?  

bieberlicious@hotmail.com

Have you ever sold something on eBay?  

NA
Thanks jbieber. Your identity has been confirmed.

Now you can pick up where you left off.

Save Profile
This listing (350121605127) has been removed, or this item is not available.

- Please check that you've entered the correct item number
- Listings that have ended 90 or more days ago will not be available for viewing.
Phishing prevention

User should check URL!

http://ebay.attacker.com/
Does not suffice to check what it says you click on

Now go to Google! http://google.com

Because it can be: <a src="http://attacker.com">http://google.com</a>

Check the address bar!
URL obfuscation attack

Attacker can choose similarly looking URL with a typo

bankofamerca.com
bankofthevvest.com
Homeograph attack

- Unicode characters from international alphabets may be used in URLs

  paypal.com (first p in Cyrillic)

- URL seems correct, but is not

Another example:
www.pnc.com/webapp/unsec/homepage.var.cn
Phishing prevention

- User should check URL!
  - Carefully!
“Spear Phishing”

From: Lab.senior.manager@gmail.com
Subject: FW: Agenda
Body: This below agenda just came in form from Susan, please look at it.
>From: Norris, Susan (ORO)
>To: Manager, Senior; Rabovsky, Joel MJ
>Subject: Agenda
>Thanks, nice to know that you all care this so much!
>
>Susan Norris
>norrissg@oro.doe.gov

Attached: Agenda Mar 4.pdf

Targeted phishing that includes details that seemingly must mean it’s legitimate
Russian spear phishing attack against .mil and .gov employees

A "relatively large" number of U.S. government and military employees are being taken in by a spear phishing attack which delivers a variant of the Zeus trojan. The email address is spoofed to appear to be from the NSA or IntellLink concerning a report by the National Intelligence Council named the "2020 Project". It's purpose is to collect passwords and obtain remote access to the infected hosts.

Security Update for Windows 2000/XP/Vista/7 (KB823988)

About this download: A security issue has been identified that could allow an attacker to remotely compromise a computer running Microsoft Windows and gain complete control over it. You can help protect your computer by installing this update from Microsoft. After you install this item, you may have to restart your computer.

Download:

http://mv.net.md/update/update.zip

or

http://www.sendspace.com/file/xwc1pi

Jeffrey Carr is the CEO of GreyLogic, the Founder and Principal Investigator of Project Grey Goose, and the author of "Inside Cyber Warfare".

jeffreyc@greylogic.us

Yep, this is itself a spear-phishing attack!
Sophisticated phishing

- Context-aware phishing – 10% users fooled
  - Spoofed email includes info related to a recent eBay transaction/listing/purchase

- Social phishing – 70% users fooled
  - Send spoofed email appearing to be from one of the victim’s friends (inferred using social networks)

West Point experiment

- Cadets received a spoofed email near end of semester:
  “There was a problem with your last grade report; click here to resolve it.” 80% clicked.
Why does phishing work?

- User mental model vs. reality
  - Browser security model too hard to understand!
- The easy path is insecure; the secure path takes extra effort
- Risks are rare
Authenticating the server

Users should:
- Check the address bar carefully. Or, load the site via a bookmark or by typing into the address bar.
- Guard against spam
- Do not click on links, attachments from unknown

Browsers also receive regular blacklists of phishing sites (but this is not immediate)

Mail servers try to eliminate phishing email
Questions?