Common Websites Security Issues

Ziv Perry

About me

- <short description>

Mitnick attack

Transitive trust

TCP splicing

Sql injection

XSS

Denial of Service

SYN flooding

CSRF

DNS Spoofing

ICMP bombing

Source routing

Sql injection

XSS

CSRF

XSS Cross Site Scripting

CSRF Cross Site request forgery

Hacking & Websites?

- More and more applications are porting for the Internet – some are written for on line use only
- On line commerce and services, include financial, government etc.
- No standard for digital signature for the mass
- The browser has become the most important tool in computers

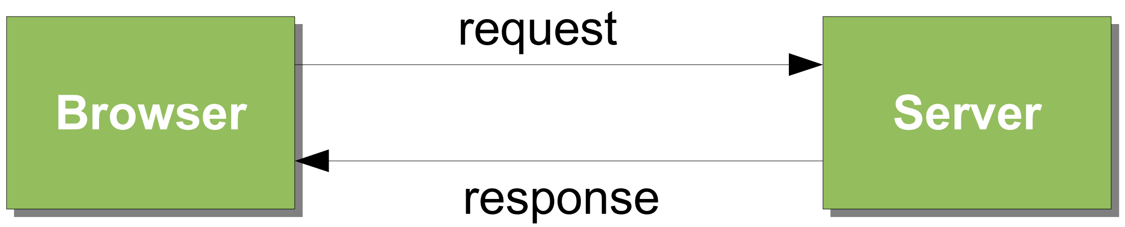
Websites security is more important than ever

Very brief introduction to Web apps

And no, we are not going to be a web developers...

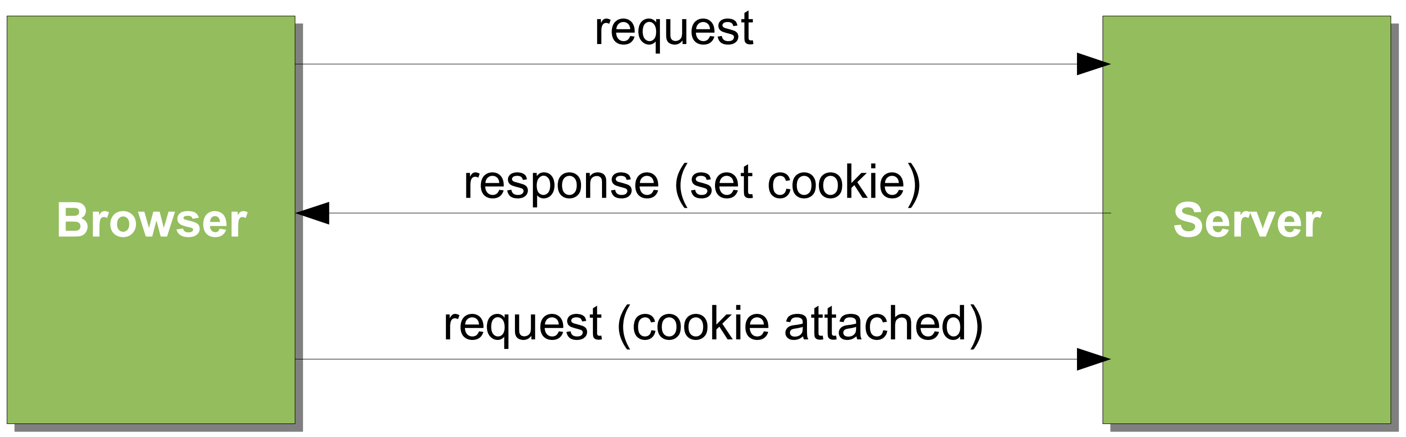
HTTP the protocol of the web

- Client/Server
- Several requests type
  - post/get/put...
- ***Stateless***
  - Has significant impact on application design and security



Facing stateless problem

- *Cookies* are small pieces of data stored in the browser and attach to *each* request
  - As a result – *cookies* “add” state



Cookies

- *Cookies* are used for:
  - Authentication
  - Personalization
  - Tracking
- *Cookies* restrictions:
  - Ownership
  - Expiration (temporaries, persistent, third-party)

Javascript

- Powerful dynamic scripting language
  - EMCAScript (EMCA-262)
- Embedded – enable programmatic access to objects within other application
  - Primarily used in web browsers for creating dynamic websites

Javascript security model

- Script run in a “sandbox”
  - No access to the OS (file system, network, etc.)
- Same-origin policy
  - Can only access to the document/window object properties from the same server (domain), protocol (scheme) and port
- User can grant privileges to signed scripts

Browser security model

- Same-origin policy
- Library import
    - Javascript from cross domain, behave as local
- Data export
    - Data can be send anywhere

XSS

Cross-Site Scripting

XSS exploit concerns the ability of a website to run scripts within the web browser, using Javascript.
Obviously the browser designed to sandbox the script, so this has restricted access to the computer running the browser.

Cross-Site Scripting

But the browser can only have low-level information to limit what the script can do.
So if the attack is at a higher conceptual level of abstraction, the lower level of logic at which the browser sandboxing of website delivered scripts occurs will not be effective.

XSS risks

- “reflection attack”
  - User is tricked to visit buggy (badly written) site
  - The browser run the attack script
- Sending users private data to the attacker
  - Cookies data, form data, keystrokes, etc.
- Changing content/behavior of a website
  - Fake user actions, fishing, disguise

XSS Attack Types

Theory and Practice

Type 0: DOM-based (local)

- Local attack
- Occurs in a context that the web browser treats as of *local origin*, allowing for unprivileged access to local objects
- Persistent & non persistent
- Cross-Zone Scripting

Type 1: Non persistent (reflected)

- Arises when an attacker succeeds in getting a victim to click on a supplied URL which is then submitted to another website.
- Occurs when server side pages are generated from client side input
- Most popular attack

XSS type 1 example *(live sample)*

- Simple “Hello user!” form:

```
Enter your name: <input type="text" name="username" />
<input type="submit" value="GO" />
```

```
<p>Hello <?php echo $_GET['username']; ?>!</p>
```

- GET /?username=<script>alert(1);</script>

```
<p>Hello <script>alert(1);</script></p>
```

Type 2: Persistent (stored)

- Malicious data stored on web server
  - websites allow inserting HTML content
- Most potentially harmful attacks
  - Attack anyone who enter this website

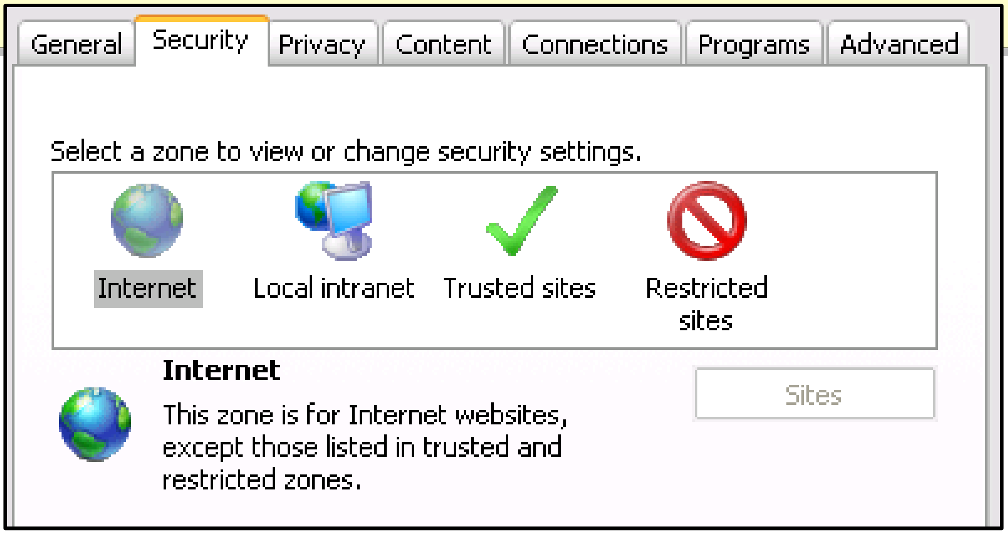
XSS type 2 example *(live sample)*

- Malicious data stored in server (article, forum post, blog comment etc.)
- The malicious script is execute every time the page displayed

XSS type 0 example (ie)

Attacker use the privileges granted by *local zone* to access file system and applications

```
<a href="C:\WINDOWS\Help\ciquery.htm?[XSS_ATTACK]”>link</a>
```



Select a zone to view or change security settings.

Internet

This zone is for Internet websites, except those listed in trusted and restricted zones.

XSS live example (or VIDEO)

This is a restricted slide until the website with the XSS exploit will fix the problem or till 24.5.09 (the sooner)

Avoiding XSS

- *Never trust user input!*
- If there is no reason to, never allow HTML in user input.
- Escape all characters (HTML entities)

```
<script> alert("1"); </script>
```

```
&lt;script&gt; alert(&quot;1&quot;); &lt;/script&gt;
```

Avoiding XSS on HTML user input

- *Never trust user input!*
- Remove all scripts tags
- Remove all DOM events from HTML tags

```
<div ~~onclick~~="foo();">
   <p ~~onmouseover~~="bar();">
```

- Filter all content for known XSS exploits
  - http://ha.ckers.org/xssAttacks.xml

XSRF

Cross-Site Request Forgery

- XSRF, also known as one click attack or session riding.
- XSRF exploits the trust a website has in a user by forging a request from a trusted user.
- These attacks are often less popular, more difficult to defend against than XSS attacks, and, therefore, more dangerous.

Creating a forgery request

- Hyperlink (require user interaction)

```
<a href="http://mybank.com?action=transfer...">link</a>
```

- HTML tags

```
<img src="http://mybank.com?action=tranfer..." />
```

```
<script src="http://mybank.com?action=tr..."></script>
```

```
<iframe src="http://mybank.com?action=tr..." ></iframe>
```

Creating a forgery request

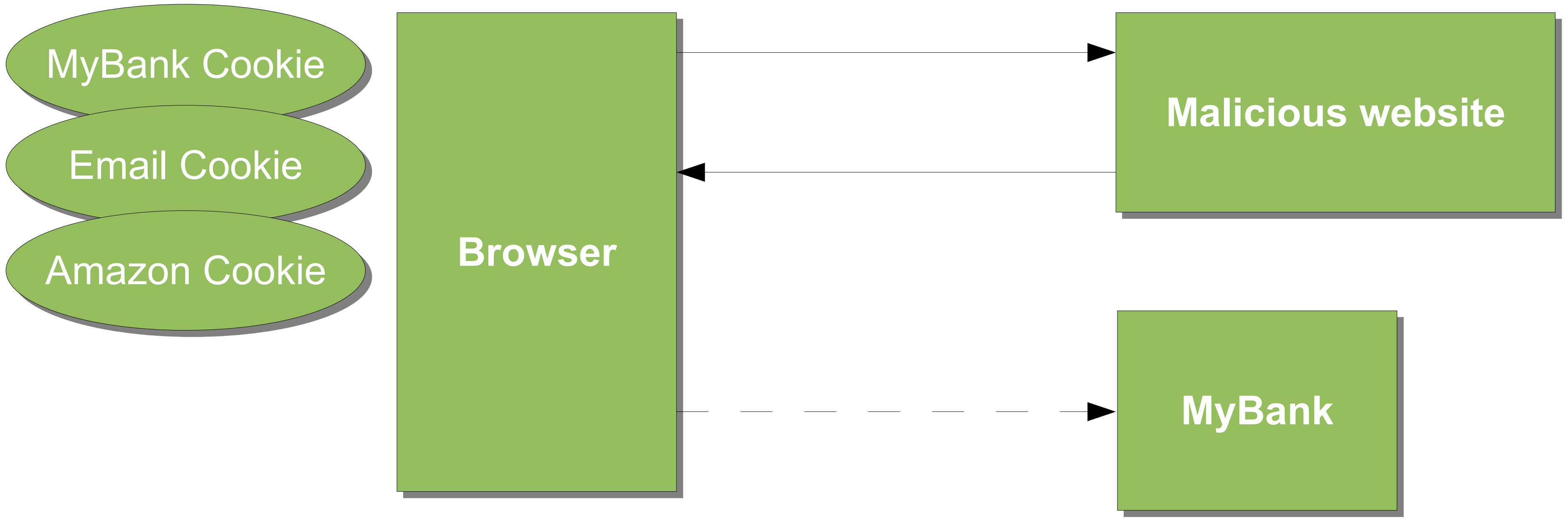
- CSS

```
.xsrf  {
   background:url("http://mybank.com?...");
}
```

- Javascript (and Flash)
  - Using Javascript we can manipulate the DOM and create any tag (with *src* attribute for GET requests) or even fully functional forms and submitting them (for POST requests)

XSRF example *(live sample)*

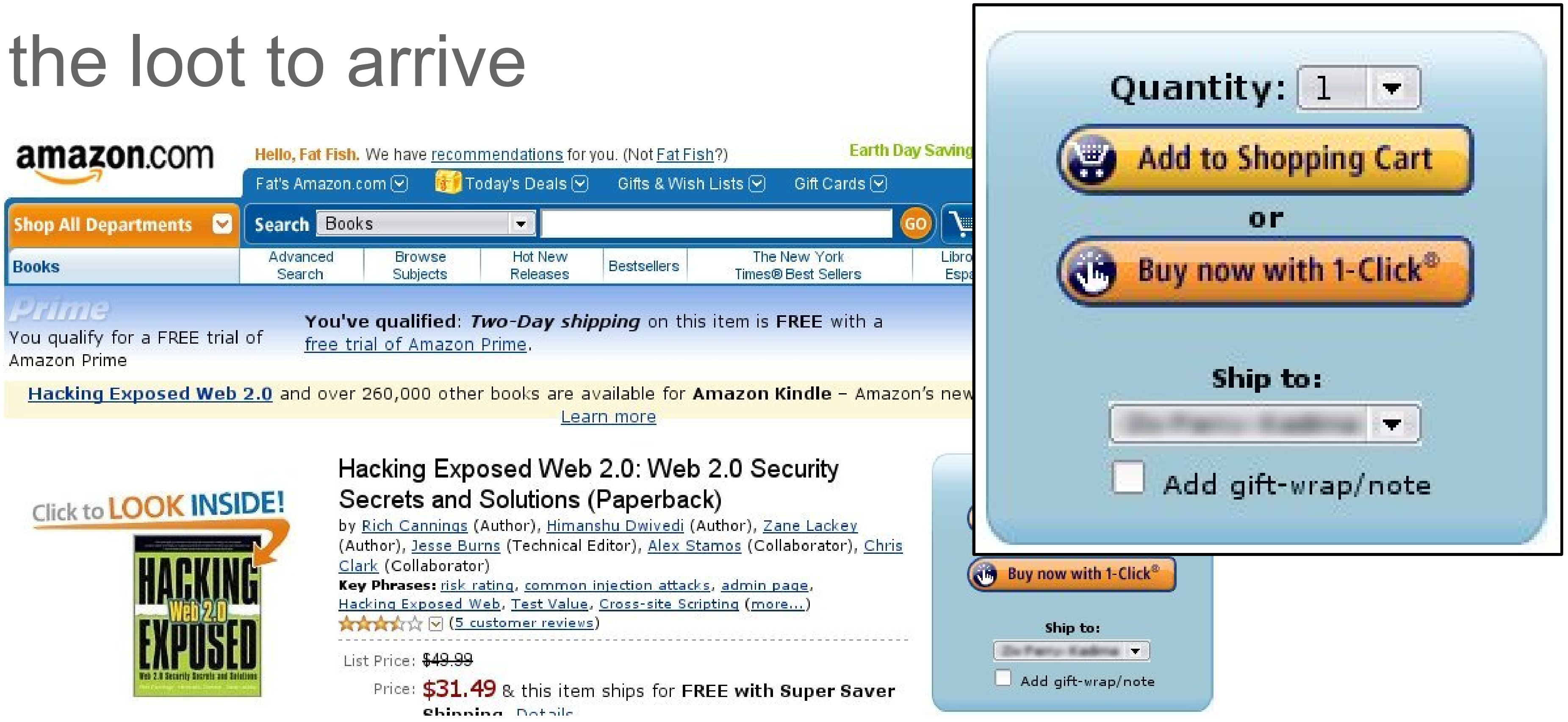
- The attack works by including a link, script or any *request-tag* in a page that accesses a site to which the user is known to have authenticated.



XSRF Amazon 1-Click example

Shopping for free at amazon using XSRF exploit

Attacker create a forgery request and just wait for the loot to arrive



amazon.com Hello, Fat Fish. We have recommendations for you. (Not Fat Fish?) Earth Day Saving

Fat's Amazon.com | Today's Deals | Gifts & Wish Lists | Gift Cards

Shop All Departments | Search Books | GO

Books | Advanced Search | Browse Subjects | Hot New Releases | Bestsellers | The New York Times® Best Sellers | Libro Espa

Prime
You qualify for a FREE trial of Amazon Prime

You've qualified: *Two-Day shipping* on this item is **FREE** with a free trial of Amazon Prime.

Hacking Exposed Web 2.0 and over 260,000 other books are available for **Amazon Kindle** – Amazon's new
Learn more

Click to LOOK INSIDE!

Hacking Exposed Web 2.0: Web 2.0 Security Secrets and Solutions (Paperback)
by Rich Cannings (Author), Himanshu Dwivedi (Author), Zane Lackey (Author), Jesse Burns (Technical Editor), Alex Stamos (Collaborator), Chris Clark (Collaborator)
Key Phrases: risk rating, common injection attacks, admin page, Hacking Exposed Web, Test Value, Cross-site Scripting (more...)
(5 customer reviews)

List Price: ~~$49.99~~
Price: $31.49 & this item ships for **FREE with Super Saver Shipping**. Details

Quantity: 1
Add to Shopping Cart
or
Buy now with 1-Click®

Ship to:
[redacted]
Add gift-wrap/note

Buy now with 1-Click®
Ship to:
[redacted]
Add gift-wrap/note

XSRF Defenses – Salting forms

- Using a unique token to identify the request

```
<form method="post">
   <input type="hidden" name="salt" value="<TOKEN>" />
   ...
</form>
```

- TOKEN =>hash(user_id + salt) + salt

XSRF Defenses – Referer check

- Checking HTTP referer header against authorized actions/pages list (not only domain)

```
http://www.example.org/manage/deleteUser?userId=12

GET /manage/deleteUser?userId=12 HTTP/1.1
Host: www.example.org
User-Agent: Mozilla/5.0 ...
Accept: text/html,application/xhtml+xml ...
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7 ...
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.malicious.com/fromPage
```

XSRF Defenses - Auth & Approval

- By creating an approval page for actions, the one-click attack is eliminated
- Re-Authorization is the same method, with an extra protection against stolen identity

XSS vulnerabilities bypass all XSRF protections

Thank You!

The slides is available at: (insert link here)

Ziv Perry
info@zivperry.com

Further reading

OWASP the free and open application security community
http://www.owasp.org

The Cross-Site Scripting (XSS) FAQ
http://www.cgisecurity.com/xss-faq.html

The Cross-Site Request Forgery (CSRF/XSRF) FAQ
http://www.cgisecurity.com/csrf-faq.html

Peter Watkins discovers Client-Side Trojans
http://www.tux.org/~peterw/csrf.txt

CERT® Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests
http://www.cert.org/advisories/CA-2000-02.html

Thomas Schreiber discovers CSRF
http://www.securenet.de/papers/Session_Riding.p

Jesse Burns discovers CSRF
http://www.isecpartners.com/files/XSRF_Paper_0

Cross-site scripting
http://en.wikipedia.org/wiki/Cross-site_scripting

Cross-site request forgery
http://en.wikipedia.org/wiki/Cross-site_request_fo