WEAPONIZING THE WEB
MORE ATTACKS ON USER GENERATED CONTENT
CITIZEN: NATHAN HAMIEL
Senior Consultant - Idea InfoSec
Associate Prof @UAT, Hexagon Security Group
23rd Degree Mason, LavaRolling Enthusiast

CITIZEN: SHAWN MOYER
Principal Consultant - FishNet Security
Douchebag with microphone, self-styled Wikipedian
Shot a man in Reno just to watch him die
★ Navel gazing and rants
★ Democratization of misinformation
★ Trust, integration, and shared exposure
★ Features arms race, emerging attack surface

★ Actual information and content
★ A nifty (we think) approach to an old bug
★ Tool release, ensuing demos o' fail
★ Stupid API tricks and multi-site mayhem
★ Sorry, you have to listen to rants first. =)
Voice of the People

- User-Generated Content
  - User-driven, social, collaborative content
  - Blogs, wikis, socnets, web communities
  - Increasingly bolted onto “old” web media

- Integrated, Aggregated, Dynamic
  - Offsite content, syndication, shared APIs
  - Aggregation points, feeds, personal portals
  - Increasing client-side logic (REST, JSON, etc)
Moot is Time's person the year
★ Lulzy example. Larger problem.
★ Time: “Feh. Internet polls aren't trusted.” Oh.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Avg. Rating</th>
<th>Total Vote</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Moot</td>
<td>87</td>
<td>12,939,521</td>
</tr>
<tr>
<td>2</td>
<td>Ahwar Ibrahim</td>
<td>42</td>
<td>1,632,411</td>
</tr>
<tr>
<td>3</td>
<td>Rick Warren</td>
<td>42</td>
<td>1,290,988</td>
</tr>
<tr>
<td>4</td>
<td>Baitullah Mehsud</td>
<td>40</td>
<td>1,281,854</td>
</tr>
<tr>
<td>5</td>
<td>Larry Brilliant</td>
<td>39</td>
<td>1,425,061</td>
</tr>
<tr>
<td>6</td>
<td>Eric Holder</td>
<td>38</td>
<td>1,215,008</td>
</tr>
<tr>
<td>7</td>
<td>Carlos Slim</td>
<td>37</td>
<td>1,311,525</td>
</tr>
<tr>
<td>8</td>
<td>Angela Merkel</td>
<td>37</td>
<td>1,069,787</td>
</tr>
<tr>
<td>9</td>
<td>Kobe Bryant</td>
<td>36</td>
<td>1,195,005</td>
</tr>
<tr>
<td>10</td>
<td>Evo Morales</td>
<td>34</td>
<td>1,045,245</td>
</tr>
<tr>
<td>11</td>
<td>Alexander Lebedev</td>
<td>34</td>
<td>640,115</td>
</tr>
<tr>
<td>12</td>
<td>Lil’ Wayne</td>
<td>33</td>
<td>637,426</td>
</tr>
<tr>
<td>13</td>
<td>Sheikh Ahmed bin Zayed Al Nahyan</td>
<td>32</td>
<td>622,054</td>
</tr>
<tr>
<td>14</td>
<td>Odell Barnes</td>
<td>31</td>
<td>621,182</td>
</tr>
<tr>
<td>15</td>
<td>Tina Fey</td>
<td>30</td>
<td>646,446</td>
</tr>
<tr>
<td>16</td>
<td>Hu Jintao</td>
<td>29</td>
<td>614,359</td>
</tr>
<tr>
<td>17</td>
<td>Eric Cantor</td>
<td>28</td>
<td>580,189</td>
</tr>
<tr>
<td>18</td>
<td>Gamal Mubarak</td>
<td>27</td>
<td>580,389</td>
</tr>
<tr>
<td>19</td>
<td>Ali al-Naimi</td>
<td>26</td>
<td>627,786</td>
</tr>
<tr>
<td>20</td>
<td>Muqtada al-Sadr</td>
<td>25</td>
<td>564,094</td>
</tr>
<tr>
<td>21</td>
<td>Elizabeth Warren</td>
<td>24</td>
<td>559,800</td>
</tr>
<tr>
<td>22</td>
<td>Nancy Pelosi</td>
<td>23</td>
<td>488,094</td>
</tr>
</tbody>
</table>
WHAT COULD POSSIBLY GO WRONG?

★ Post-MJ celebrity death hoaxes
★ Some “real” news outlets picked up.
★ iReport, uReport, you are on notice.
★ Note: Please stop Rickrolling. Please.

1980s pop icon Rick Astley, 43, found dead in Berlin hotel room

Published: Tuesday June 30, 2009 MYT 2:48:00 PM

BERLIN: Known for his 1980s pop hit Never Gonna Give You Up, 43-year-old Rick Astley has been pronounced dead Tuesday.

His body was found at the Angleterre Hotel in Berlin after an ambulance responded to an emergency call from his hotel room.
**WHAT COULD POSSIBLY GO WRONG?**

- NYT aggregation fail
  - HTML injection article propagates HTML injection
  - Aggregation, syndication, shared exposure
What could possibly go wrong?

- DailyKos trolls twittering dittoheads
- Fake economy / budget numbers
  - $750,000 for an underground tunnel connecting a middle school and high school in North Carolina.
  - $4.7 million for a program supplying public television to K-8 classrooms.
  - $2.3 million for a museum dedicated to the electric bass guitar.
The emerging socialized web
- Multi-site aggregation = Attacker ROI
- Multipoint attack surfaces, APIs, “Digg this!”, etc
- (n)th-parties and shared exposure

“Malware-like” legit functionality
- Silent updates, presence announcements
- Offsite links and wrapped external content
- Try blocking .js for googleapis.com. I dare you.
**File Sharing**
A simple and safe way to share files directly from your computer.

**Photo Sharing**
Share your personal photos with friends around the world without the need to upload them.

**Fridge**
A fun place for people to leave notes on your computer.

**The Lounge**
Invite your friends to a chat in The Lounge hosted on your computer.

**Media Player**
Access your complete home music library from wherever you are.

**Web Server**
Host your Web sites running from your own computer.
Top Botsites

As you browse the web, Safari will learn which websites are your favorites, and replace the websites above with those websites.
Retrofitting the Thing of The Now

More FF fail. No, srsly.
APIs are the New Hotness

- Integrate other site functions (*Your* tweets in *my* Facebook? Awww....)
- Hooks into fluffy clouds of amorphous love
  - googleapis, amazonws, others
  - Crossdomain content, sandboxing

Two major types of APIs

- For consumption of application services
- For integration of app on another site
Your app is so ugly its APIs have APIs

How far away from what we are using do we need to be?

★ = WTF. Complexity breeds exposure.
Attacks anonymization via shared APIs
Hi5 API localhost dev page. Opps1!1
Triangle of Death

(Rectangle|Pentagon|Hexagram|Octagon) of Death
CSRF / Session Riding / XSRF

- Well understood. Pete Watkins, 2001
- Often tough to audit for, nuanced
- Typically described as a “static” attack
- Per-user forgeries *usually* only via XSS

- Can be silly, bad, or really, really bad
  - Our continued move to webeverything\(^{\text{tm}}\)
  - Classical mitigations: Referrer, POSTs, tokens
Do you use a browser for it?
CLASSICAL CSRF

Example:

```
<img src="http://good.com/poll.php?poll=5&selection=2" height="1" width="1"/>
```
CLASSICAL CSRF (VIA POST)

The Recession Is Over. Now What?

Wall Street is up. The housing market is improving. But growth is weak, and, most important, jobs are still scarce. What we need is a new kind of recovery.

The Rough Road to Recovery

The Sex Offenders Under the Bridge

The rough road to recovery, by Daniel Gross

Our Man in Afghanistan

Richard Holbrooke’s impossible mission

America’s Newest Nightmare

Meet the Taliban’s next leader

BLACK HAT USA 2009
“Dynamic” CSRF

★ "Dynamic" CSRF.
★ Per-request, per-session, per-user forgeries
★ Watkins described in 2001, but no one noticed
★ Samy, recent bit.ly XSS, other XSS worms
★ Again, well understood as XSS side effect

★ Lots of "complex" CSRF gets ignored
★ POST-based, tokenized, per-user requests
★ Still exploitable, but higher bar
★ `<img src="/password?newpassword=moo">` gets old after the 30 times or so.
“Dynamic” CSRF.

- We wanted to automate “complex” CSRF
- Needed more logic than just redirects / tags
- Many non-trivial CSRF are ignored
  - Devs often think SOP saves them (it might)

Dynamic CSRF

3-Way Site Communication
1. Initial Request
2. Redirect to bad.com
3. Custom payload for site

Custom Payload for site w/ tokens, session IDs, etc.

Redirected Request w/ referer, CSRF tokens, session IDs, etc.

good.com

kindagood.com

sortagood.com

bad.com
MonkeyFist: PoC Dynamic CSRF Tool

- http://hexsec.com/labs
- Small Python web server
- Creates payload / patterns based on referrer
- Automates per-request, “dynamic” CSRF
- Constructs hidden POSTs, redirects, refreshes
- Makes requests for tokens or steals from referrer
MF Payload Options

★ `<PAYLOAD n="1">` - Payload with number
★ `<SITE l="example.com>` - Site entry w/ domain
★ `<METHOD>` - Attack method (GET, POST, PAGE)
★ `<ID>` - Session data to grab
★ `<TARGET>` - URL to send attack to
★ `<HEADER>` - Header to add to POST request
★ `<HEADVAL>` - Value for defined header
★ `<POSTVAR>` - POST Variable name
★ `<POSTVAL>` - Value for defined POST variable
★ `<DESTINATION>` - Destination for meta refresh
POST CONSTRUCT

1. Request for Content
2. Cross-Domain Data
3. POST Request/Session Data

good.com

Host Making POST

Saturday, August 1, 2009
Cross-site request forgery

From Wikipedia, the free encyclopedia

Cross-site request forgery, also known as a one-click attack or session riding and abbreviated as CSRF [see note] or XSRF, is a type of malicious exploit of a web application whereby unauthorized commands are transmitted from a user that the website trusts. Unlike cross-site scripting (XSS), which exploits the trust a user has for a particular site, CSRF exploits the trust that a site has in a user's browser.

Contents [hide]
1 Background
2 Examples and characteristics
3 Limitations
4 Forging logic requests
5 Prevention
6 See also
7 References
8 External links

Background

CSRF vulnerabilities have been known and in some cases exploited since the 1990s.[3] Because it is carried out from the user's IP address, some web site logs might not have evidence of CSRF.[3] Exploits are under-reported, at least publicly, and as of 2008, there are few well-documented examples. About 13 million users of eBay's Internet Auction site at Auction.ca in Korea lost personal information in February 2006.[6] Customers of a bank in Mexico were attacked in early 2000 with an image tag in email and were sent through their home routers to the wrong website.[8]

Example and characteristics

---

Saturday, August 1, 2009
★ MF “Dynamic” CSRF of anon Wikipedia edit
★ Requests were replayable, but unique
★ WPEdittime, WPStarttime, other session values
★ MF requested session values, hidden POST
★ We think this is pretty nifty.

OMGTHETANS!
<PAYLOAD n="5">
    <SITE i="stlouis.craigslist.org">
        <METHOD>FIXATION</METHOD>
        <DESTINATION>http://www.youtube.com/watch?v=ZAlNoO0oaNw</DESTINATION>
        <FIXVAR>wpStarttime</FIXVAR>
        <FIXVAL>wpStarttime</FIXVAL>
        <FIXVAR>wpEdittime</FIXVAR>
        <FIXVAL>wpEdittime</FIXVAL>
        <FIXVAR>wpAutoSummary</FIXVAR>
        <FIXVAL>wpAutoSummary</FIXVAL>
        <POSTVAR>wpAntispam</POSTVAR>
        <POSTVAL></POSTVAR>
        <POSTVAR>wpSection</POSTVAR>
        <POSTVAL></POSTVAR>
        <POSTVAR>4</POSTVAR>
        <POSTVAR>wpScrolltop</POSTVAR>
        <POSTVAR>0</POSTVAR>
        <POSTVAR>wpSummary</POSTVAR>
        <POSTVAR></POSTVAR>
        <POSTVAR>wpSave</POSTVAR>
        <POSTVAR>Save+page</POSTVAR>
        <POSTVAR>wpEditToken</POSTVAR>
        <POSTVAR>+</POSTVAR>
    </SITE>
</PAYLOAD>
CSRF mitigations are well understood
Still, you have to LOTS of things right
No bolt on fixes, sorry.
Look at your code! Forget SOP.
Thanks for listening. Send bugfixes.
Nathan’s blog: http://www.neohaxor.org
Shawn hates blogs.