



# Vulnerability Analysis, Secure Development and Risk Management of Web 2.0 Applications

Marco Morana  
OWASP

**OWASP**  
Cincinnati Chapter,  
November 2010  
Meeting

Copyright © 2010 - The OWASP Foundation  
Permission is granted to copy, distribute and/or modify this document  
under the terms of the GNU Free Documentation License.

**The OWASP Foundation**  
<http://www.owasp.org>

# What is OWASP?



# OWASP

The Open Web Application Security Project

<http://www.owasp.org>

Organization Supporters of OWASP's mission

ACCUVANT LABS

acunetix



ASPECT SECURITY  
Application Security Specialists

art defence



Booz | Allen | Hamilton

BEST BUY

BREACH

cigital



CorporateOne

DENIM GROUP

DREAMLAB  
TECHNOLOGIES

ETS

Listening. Learning. Leading.

fishnet  
SECURITY

FUJITSU  
THE POSSIBILITIES ARE INFINITE

GO THAM  
DIGITAL SCIENCE

HARRIS  
CONNECT



IMPERVA

Information  
Builders

INFOVISION

(ISC)<sup>2</sup>

Microsoft

mnemonic

NOKIA

salesforce.com  
Success On Demand

securestate

protiviti  
Independent Risk Consulting

SOGETI

symantec

WhiteHat  
SECURITY

OWASP



# Agenda For Today's Presentation

1. The Evolution of Web 2.0
2. Web 2.0 Vulnerability Analysis
3. Building Secure Web 2.0 Applications
4. Web 2.0 Risk Management

# **The Evolution of the Internet to Web 2.0**

# General Web 2.0 Background

- **Can be defined as: “Web applications that facilitate interactive information sharing and collaboration, interoperability, and user-centered design on the World Wide Web”**

... the main characteristics of web 2.0 are:

- 1. Encourage user’s participation and collaboration** through a virtual community of social networks/sites. Users can and add and update their own content, examples include Twitter and social networks such as Facebook, Myspace, LinkedIn, YouTube
- 2. Transcend from the technology/frameworks used** AJAX, Adobe AIR, Flash, Flex, Dojo, Google Gears and others
- 3. Combine and aggregate data and functionality from different applications and systems**, example include “mashups” as aggregators of client functionality provided by different in-house developed and/or third party services (e.g. web services, SaaS)

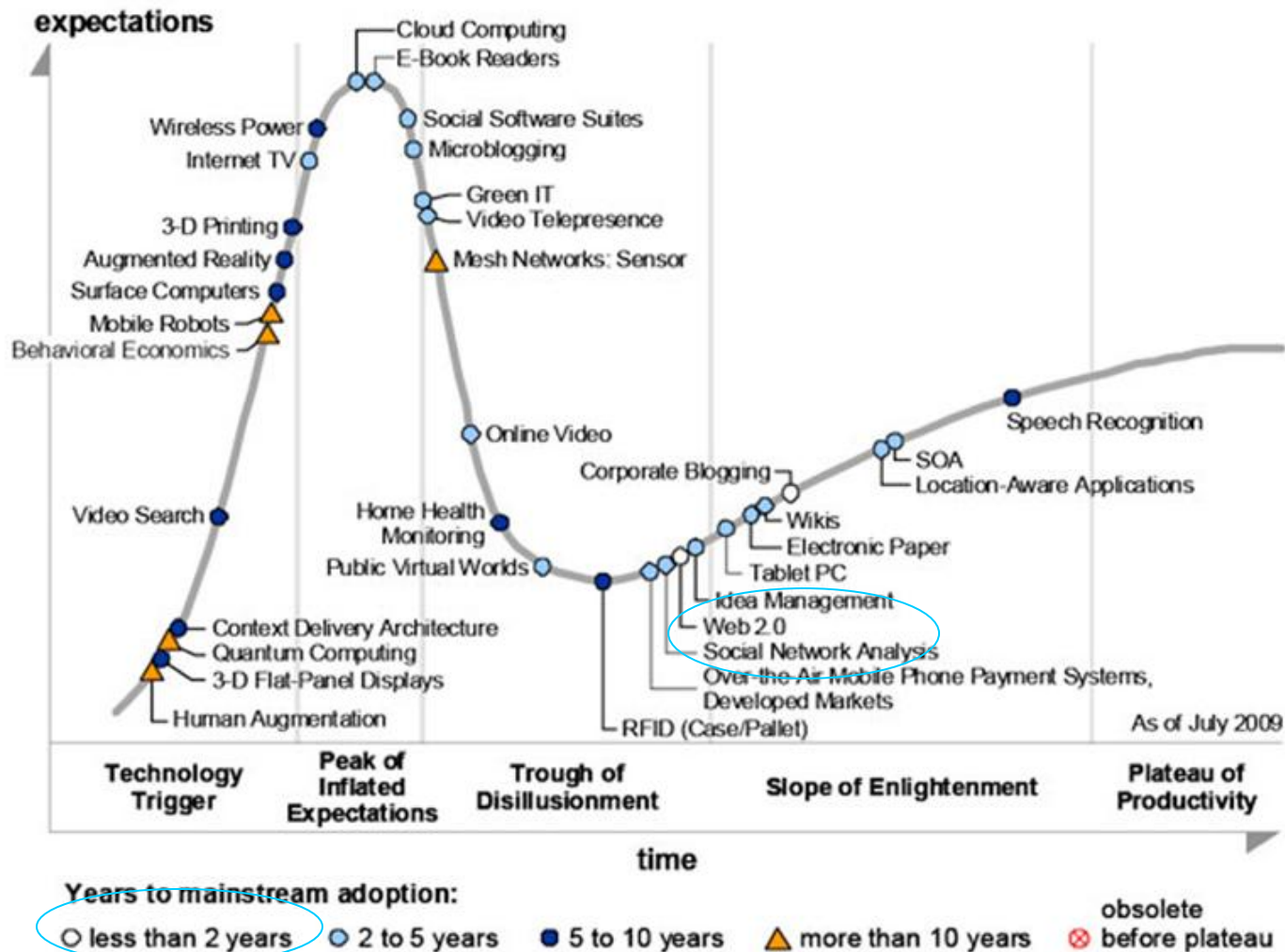
# Web 2.0 As Evolution of Human Knowledge



Source <http://digigogy.blogspot.com/2009/02/digital-blooms-visual.html>

# Web 2.0 As Adoption By Businesses

Figure 1. Hype Cycle for Emerging Technologies, 2009



Source: Gartner (July 2009)

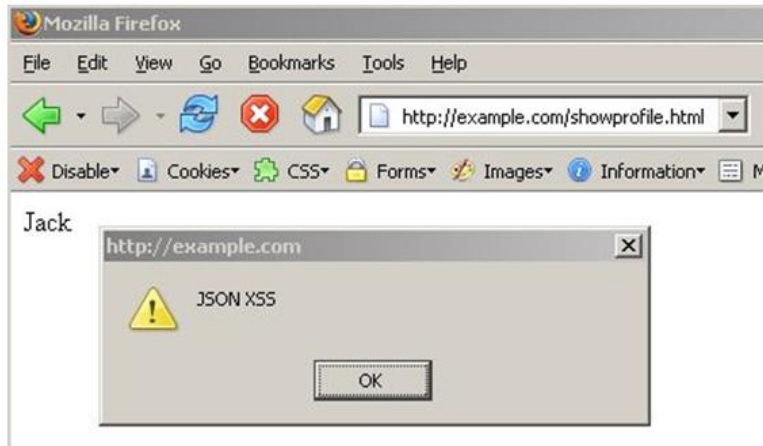


# How Web 2.0 Changes The Threat Landscape

- **Web 1.0 threats are amplified by the intrinsic nature of Web 2.0** such as expanded interaction model and use of both old and new Web 2.0 technologies, examples:
  - ▶ **Social networks as target for attack users** with malware, FaceBook is 350 Million users !
  - ▶ **Web 2.0 prone to Web 1.0 vulnerabilities** such as XSS, CSRF, Phishing, Injection Flaws
- **Web 2.0 enable more effective attacks** because of sharing and integration between disparate systems, examples are:
  - ▶ **Complexity of integration** of different technologies and services, front-end/client and back-end/server
  - ▶ **Rich client interfaces increase the attack surface** and the likelihood of business logic attacks
- **Social networks facilitate information disclosure** of confidential PII, examples are:
  - ▶ **Abuse of user's trust first-verify model by attackers**
  - ▶ **Sharing data model breaks boundaries of confidentiality**, not clear boundaries between private vs. public, personal life vs. professional life



# Web 2.0: Old Vulnerabilities And New Exploit Scenarios



## THREAT LEVEL

PRIVACY, CRIME AND SECURITY ONLINE

PREVIOUS POST

NEXT POST

### Hackers Use Twitter to Control Botnet

By Ryan Singel | August 13, 2009 | 7:34 pm | Categories: Crime, Hacks and Cracks

Hackers are now using Twitter to send coded update messages to computers they've previously infected with rogue code, according to a report from net-monitoring firm Arbor Networks.

This looks to be the first reported case of hackers using the popular micro-messaging company to control botnets, which are assemblages of infected PCs that can be directed to spy on their users, send spam, or attack web sites with fake traffic.

Arbor Network's Jose Nazario, an expert on botnets, discovered the so-called command-and-control structure. Infected computers were following the Twitter feed "Upd4t3" (now suspended) through its RSS feed.



# Web 2.0 Vulnerabilities

# Top 50 WASC Threats and Top 10 OWASP Risks Especially Impacting Web 2.0

WASC Threat Classification v2	OWASP Top Ten 2010 RC1
WASC-19 SQL Injection	A1 - Injection
WASC-23 XML Injection	
WASC-28 Null Byte Injection	
WASC-29 LDAP Injection	
WASC-30 Mail Command Injection	
WASC-31 OS Commanding	
WASC-39 XPath Injection	
WASC-46 XQuery Injection	
WASC-08 Cross-Site Scripting	A2 - Cross Site Scripting (XSS)
WASC-01 Insufficient Authentication	A3 - Broken Authentication and Session
WASC-18 Credential/Session Prediction	
WASC-37 Session Fixation	
WASC-47 Insufficient Session Expiration	
WASC-01 Insufficient Authentication	A4 - Insecure Direct Object References
WASC-02 Insufficient Authorization	
WASC-33 Path Traversal	
WASC-09 Cross-site Request Forgery	A5 - Cross-Site Request Forgery
WASC-14 Server Misconfiguration	A6 - Security Misconfiguration
WASC-15 Application Misconfiguration	
WASC-02 Insufficient Authorization	A7 - Failure to Restrict URL Access
WASC-10 Denial of Service	
WASC-11 Brute Force	
WASC-21 Insufficient Anti-automation	
WASC-34 Predictable Resource Location	
WASC-38 URL Redirector Abuse	A8 - Unvalidated Redirects and Forwards
WASC-50 Insufficient Data Protection	A9 - Insecure Cryptographic Storage
WASC-04 Insufficient Transport Layer Protection	A10 - Insufficient Transport Layer Protection

# WASC-23 XML INJECTION, WASC-29 XPATH INJECTION, OWASP A1: INJECTION FLAWS

## ■ WEB 2.0 EXPLOIT SCENARIOS:

### ▶ XML INJECTION/POISONING

- **User-supplied input is inserted into XML without sufficient validation** affecting the structure of the XML record and the tags (and not just content)

### ▶ XPATH INJECTION

- **XPath injection is an attack to alter an XML query** to achieve the attacker's goals

### ▶ JSON INJECTION

- **An attacker can force execution of malicious code by injecting malicious JavaScript code into the JSON** (JavaScript Object Notation structure) on the client.

### ▶ RSS FEED INJECTION

- **RSS feeds can consume un-trusted sources injected with XSS**

## ■ WEB 2.0 KNOWN INCIDENT EXAMPLE:

- ▶ WHID 2008-47: The Federal Suppliers Guide validates login credential in JavaScript -

# WASC-08/OWASP A2: CROSS SITE SCRIPTING (XSS)

## ■ WEB 2.0 EXPLOIT SCENARIOS:

### ▶ INSUFFICIENT LIMITS ON USER INPUT

- **Users are allowed to enter HTML data that can be potentially malicious** (e.g. while creating contents such as networks, blogs or wikis)
- **Users have extensive control over user content including unsafe HTML tags** that can be abused for XSS

### ▶ INSUFFICIENT FILTERING FOR XSS DOM

- **XSS exposure is increased for Web 2.0 especially for XSS DOM** since is used in RIA written in FLASH or Silverlight, Mashups and Widgets using DOM
- **AJAX increases the entry points for potential XSS injections**

## ■ WEB 2.0 KNOWN INCIDENT EXAMPLE:

### ▶ WHID 2008-32: Yahoo HotJobs XSS

- Hackers exploiting an XSS vulnerability on Yahoo HotJobs to steal session cookies of victims

# WASC-01: INSUFFICIENT AUTHENTICATION

## OWASP-A3: BROKEN AUTHENTICATION AND SESSION MANAGEMENT

### ■ WEB 2.0 EXPLOIT SCENARIOS:

#### ▶ WEAK PASSWORDS

- User choice of **simple-to-guess passwords** and trivial password-reminder questions set by on-line site contributors

#### ▶ CLEAR TEXT PASSWORDS

- **Password stored in AJAX Widgets/Mashups** sent and stored in clear outside the control of the host

#### ▶ INSUFFICIENT PASSWORD MANAGEMENT CONTROLS

- **Password recovery/reminders not protected from brute force attacks**

#### ▶ SINGLE-SIGN-ON DESIGN FLAWS

- Passwords stored in personalized homepage and in the **desktop widget as “autologon feature”** or in the cloud to SSO from the desktop

### ■ WEB 2.0 KNOWN INCIDENT EXAMPLE:

- ▶ WHID 2009-2: Twitter Accounts of the Famous Hacked

# WASC-09/OWASP A5: CROSS SITE REQUEST FORGERY (CSRF)

## ■ WEB 2.0 EXPLOIT SCENARIOS:

### ▶ **CSRF USING AJAX REQUESTS**

- **XHR calls enable invisible queries** of a web application by the client that user cannot visually validate for forgery

### ▶ **INSUFFICIENT BROWSER ENFORCEMENT OF SINGLE ORIGIN POLICY**

- **Desktop widgets do not have the same SOA protection as browser applications** and facilitate CSRF

### ▶ **WEAK SESSION MANAGEMENT**

- **Session expiration times are typically quite high**, increasing the risk of session base attacks such as CSRF
- **Persistent session cookies are shared by Widgets** increase the opportunities for CSRF attacks

## ■ **WEB 2.0 KNOWN INCIDENT EXAMPLE:**

- ▶ WHID 2009-4: Twitter Personal Info CSRF -By exploiting a CSRF bug in Twitter, site owners can get Twitter profiles of their visitors.



# WASC-21: INSUFFICIENT ANTI-AUTOMATION

## ■ WEB 2.0 EXPLOIT SCENARIOS:

### ‣ AUTOMATIC SPREAD OF SPAM AND PHISHING LINKS

- **Spammers can automatically post links** to increase the popularity ranking of site
- **Fraudsters can use automation to embed malicious links** such as malicious advertisements for drive by download malware attacks

### ‣ AUTOMATIC REGISTRATION OF USER ACCOUNTS

- **Scripts to automatically register web e-mail accounts** in order to authenticate to other services/applications

### ‣ AUTOMATIC EMBEDDING OF COMMANDS

- **Embedding commands for controlling botnet** using RSS feeds, social networking sites

### ‣ AUTOMATIC BUSINESS LOGIC EXPLOITS

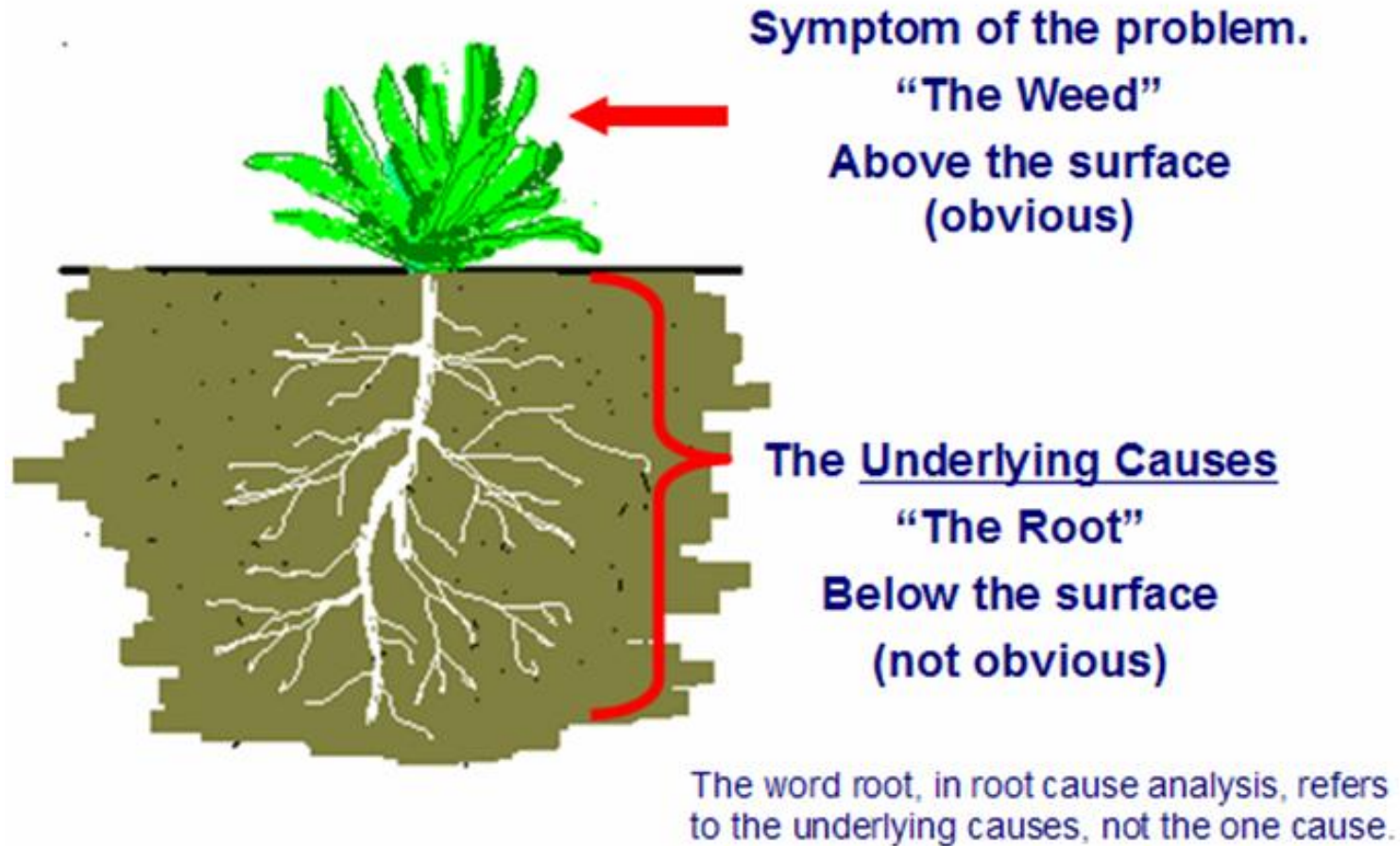
- **Automatically bid on items to increase prices**, resource exhaustion of available seats, buy and resale tickets

## ■ WEB 2.0 KNOWN INCIDENT EXAMPLE:

- WHID 2007-65: Botnet to manipulate Facebook



# Vulnerability Root Cause Analysis



# WASC Classification of Root Causes Of Web 2.0 Vulnerabilities

## 1. **USER GENERATED CONTENT**

Ability of consumers to add and update their own content

## 2. **MASHUPS & WEB SERVICES**

Aggregation of data on the desktop through mashups and web services

## 3. **DATA CONVERGENCE**

No boundary between private and public information

## 4. **DIVERSITY OF CLIENT SOFTWARE**

Data and software functions available across many different technologies and environments

## 5. **COMPLEXITY & ASYNCHRONOUS OPERATION**

Increased user interaction, integration APIs lead to complexity one of which is AJAX

# Summary of Top Web 2.0 Security Threats

VULNERABILITY	EXPLOIT SCENARIO	WEB 2.0 ROOT CAUSES
V1: INSUFFICIENT AUTHENTICATION CONTROLS	V1.1 WEAK PASSWORDS V1.2 INSUFFICIENT ANTI-BRUTE FORCE CONTROLS V1.3 CLEAR TEXT PASSWORDS V1.4 SINGLE-SIGN-ON	W1 – User contributed content W2 – Mashups, W4 – Diversity of client software, W5 - Complexity
V2: CROSS SITE SCRIPTING (XSS)	V2.1 INSUFFICIENT LIMITS ON USER INPUT	W1 – User contributed content
V3: CROSS SITE REQUEST FORGERY (CSRF)	V3.1 CREDENTIAL SHARING BETWEEN GADGETS V3.2 CSRF USING AJAX REQUESTS V3.3 LENGTHY SESSIONS	W5 - Complexity & Asynchronous Operation W2 – Mashups, W4 – Diversity of client software
V4: PHISHING	V4.1 PHONY WIDGETS V4.2 PHONY CONTENT USED FOR PHISHING V4.3 XSS EXPLOITED FOR PHISHING	W2 – Mashups, W4 – Diversity of client software W1 – User Contributed Content
V5: INFORMATION LEAKAGE	V5.1 SENSITIVE INFORMATION POSTED TO WEB 2.0 SITES V5.2 INFORMATION AGGREGATION IN SOCIAL NETWORKS V5.3 EASY RETRIEVAL OF INFORMATION THROUGH WEB SERVICES	W1 – User contributed content W3 – Consumer and enterprise worlds convergence) W4 – Mashups & Web Services
V6: INJECTION FLAWS	V6.1 XML INJECTION V6.2 XPATH INJECTION V6.3 JSON INJECTION	W4 – Mashups & Web Services, W5: Complexity & Asynchronous Operation
V7: INFORMATION INTEGRITY	V7.1 AUTHENTICATED USERS PUBLISH FRAUDULENT INFORMATION	W1 – User contributed content
V8: INSUFFICIENT ANTI-AUTOMATION	V8.1 WEB SPAM V8.2 AUTOMATIC OPENING OF USER ACCOUNTS V8.3 UNFAIR ADVANTAGE ON SITE	W1 – User contributed content W2 – Mashup & Web Services

# **Building Secure Web 2.0 Applications**

# Making Application Security Visible...



# Web 2.0 Security Engineering Essential Steps

## 1. Document Security Standards For Web 2.0

**Document Web 2.0 technology security requirements** (e.g. AJAX, FLASH) and enforce them at the beginning of the SDLC

## 2. Conduct Application Threat Modeling during design

**Examine the architecture of Web 2.0 application and all tiers for secure design** of authentication-session management, authorizations, input validation, error handling-logging

## 3. Perform Secure Code Reviews On Web 2.0 Components/Frameworks

**Assure source code adherence to security coding standards**

**Identify security bugs** in both client (e.g. Widgets, AJAX) as well as servers (e.g. Web services, SOA)

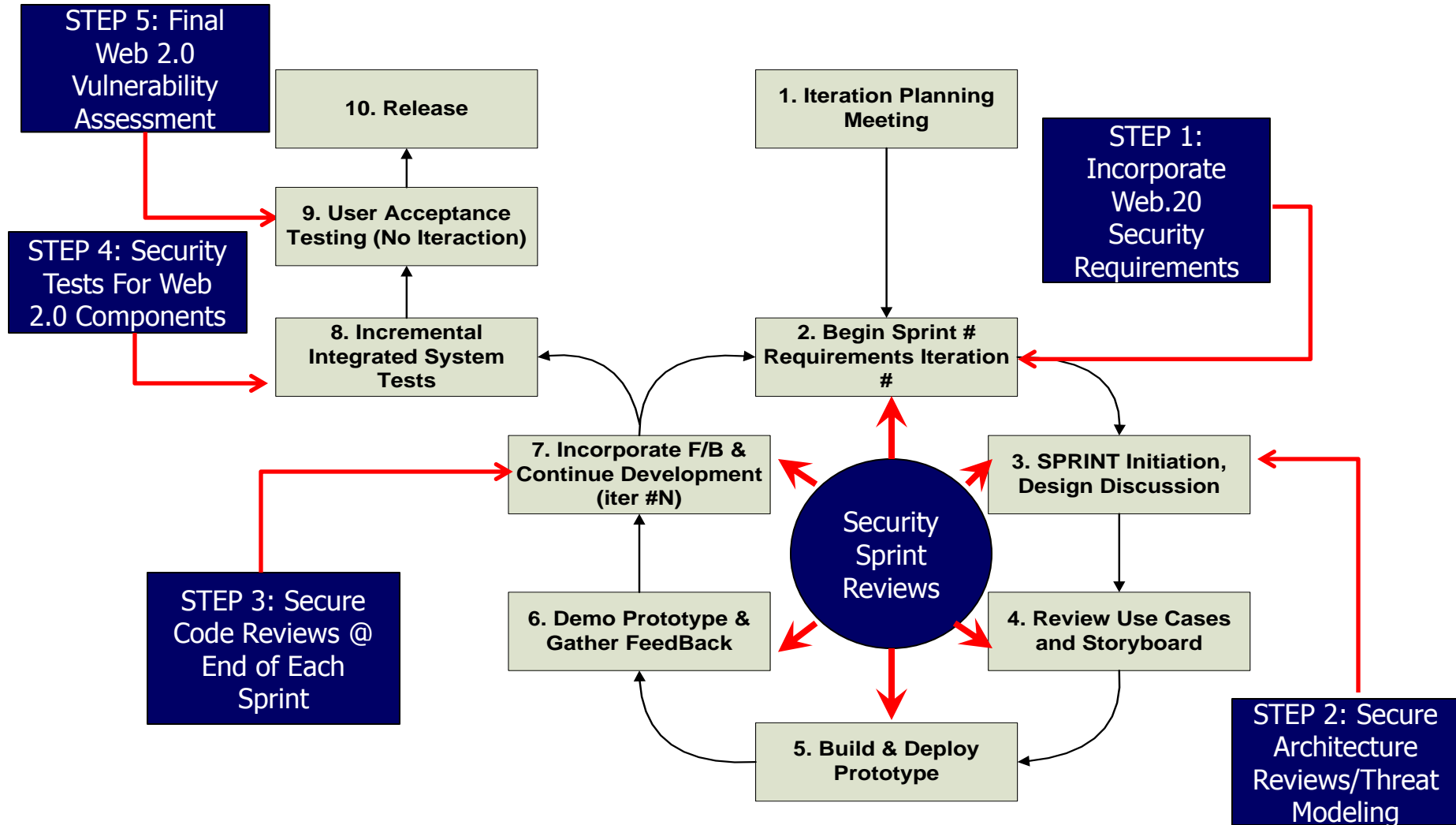
## 4. Security test Web 2.0 components

**Security test cases for AJAX and Web Services**, use the OWASP test guide test cases

## 5. Assess the whole Web. 2.0 applications for vulnerabilities

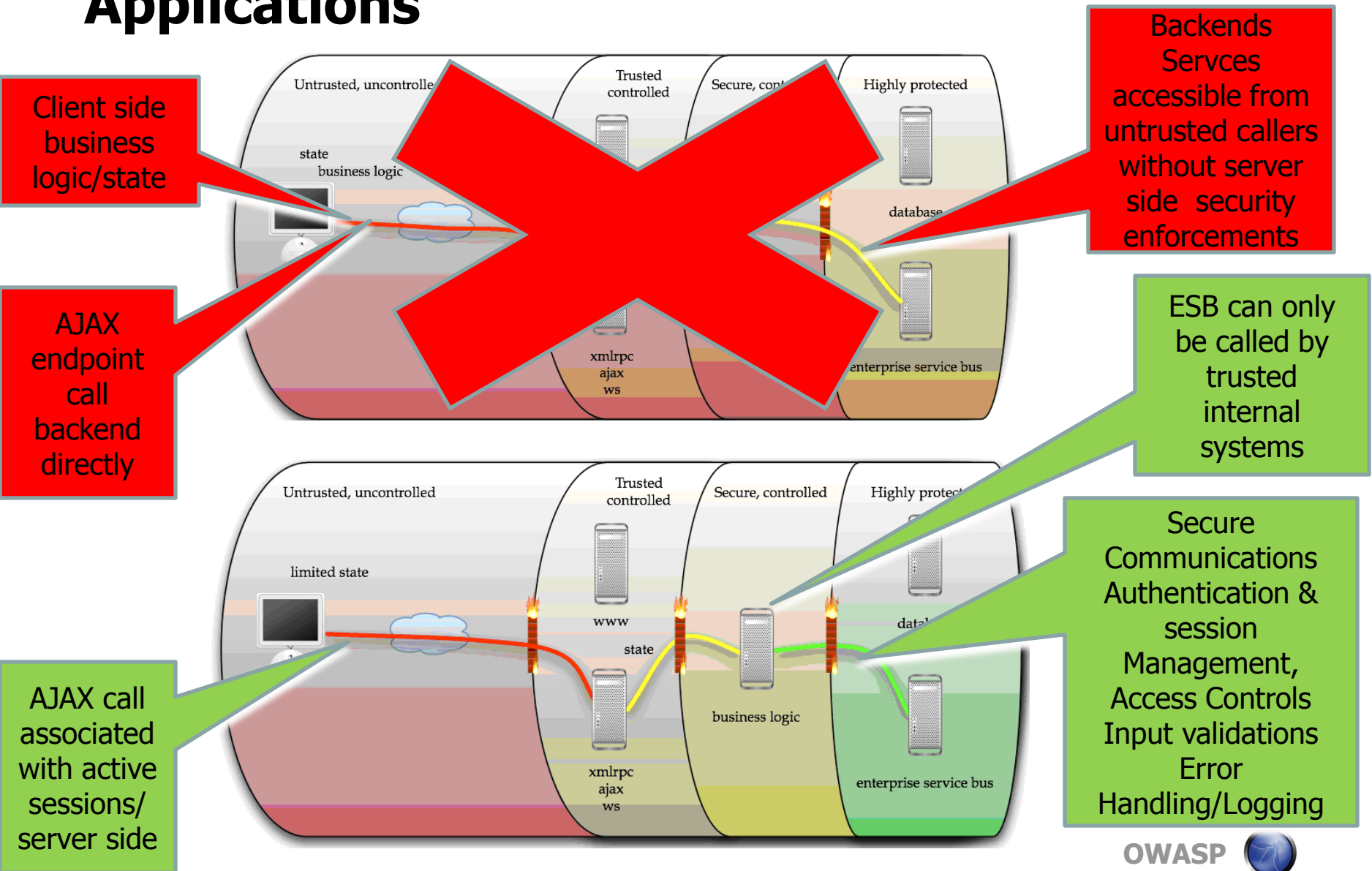
**Conduct final vulnerability assessment on whole Web 2.0 application** (e.g. test for OWASP T10, WASC, SANS-25 vulnerabilities)

# Security Touch Points For Web 2.0 using AGILE SDLC



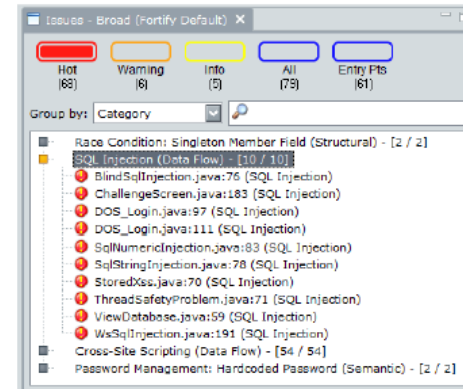
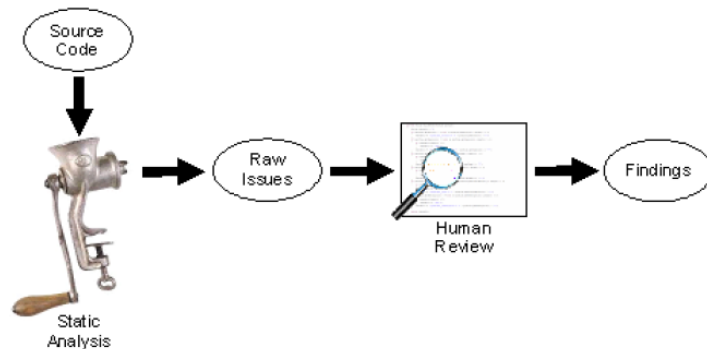


# Secure Architecting AJAX In Web 2.0 Applications





# Secure Code Reviews Of Web 2.0 Applications



OWASP CODE REVIEW  
GUIDE

2008 V1.1



WEB 2.0

Ajax and JavaScript

Look for Ajax usage, and possible JavaScript issues:

```
document.write
eval
document.cookie
window.location
document.URL
```



```
1380 /**
1381  * Set the backend
1382  *
1383  * @param object BackendObject
1384  * @throws Zend_Cache_Exception
1385  * @return void
1386  */
1387 public function setBackend(Zend_Cache_Backend BackendObject)
1388 {
1389     $this->backend = BackendObject;
1390     // Some options (listeners & directivesList) have to be given
1391     // to the backend to know if they are not 'backend specific'
1392     $directives = array();
1393     foreach (Zend_Cache_Core::getDirectivesList() as $directive) {
1394         $directives[$directive] = $this->options[$directive];
1395     }
1396     $this->backend->setDirectives($directives);
1397     // If it's a Zend_Cache_Backend_ExtendedInterface, class implements($this->backend) {}
1398     $this->extendedBackend = true;
1399     $this->backendCapabilities = $this->backend->getCapabilities();
1400 }
1401
1402 /**
1403  * Returns the backend
1404  *
1405  * @return object Backend object
1406  */
1407 public function getBackend()
1408 {
1409     return $this->backend;
1410 }
```

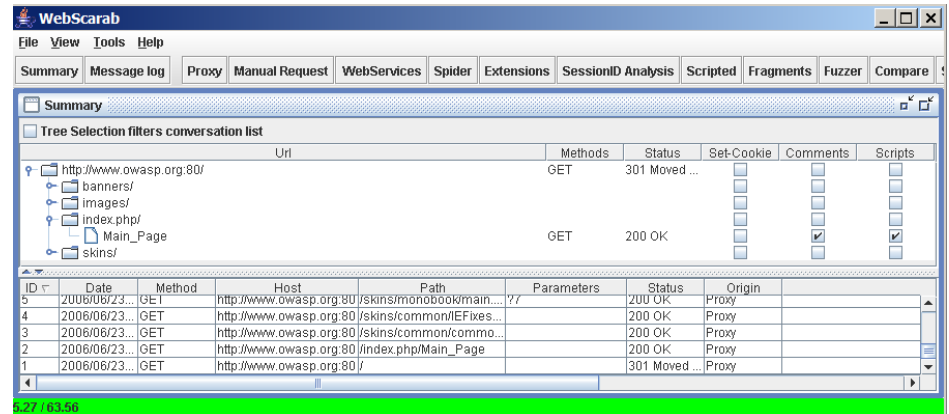
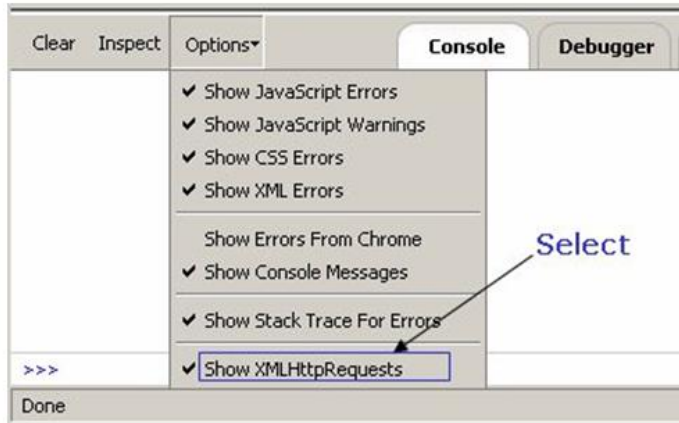
OWASP



# “TOP 10” Secure Coding Requirements for AJAX

1. **Validate data on the server side for all data entry points and URLs** of AJAX calls for code injection vulnerabilities such as Javascript injection, JSON injection, DOM injection, XML injection. Use ***JSON.parse*** to parse objects before calling eval() if used
2. **Make sure business logic is enforced on the server not by client side logic !** using server parameters
3. **Validate a well formatted XML** against allowed specification of values at server side
4. **Enforce authentication** before any XMLHttpRequest (XHR) session.
5. **Enforce authorization** checks on data accessed through XHR
6. **Add token to the URL and verify at server side for CSRF vulnerabilities** via forging of dynamic script tags.
7. **Do not store or cache sensitive data on the client** such as passwords, sessionIDs, client javascript, Flash local shared object and Mozilla's DOM storage
8. **Avoid using dynamic <script> tags** since there is no opportunity for data validation before execution
9. **Always use POST method** to send request as default
10. **Do not use javascript alert()** for error handling

# Secure Testing Web 2.0 Client and Server Components

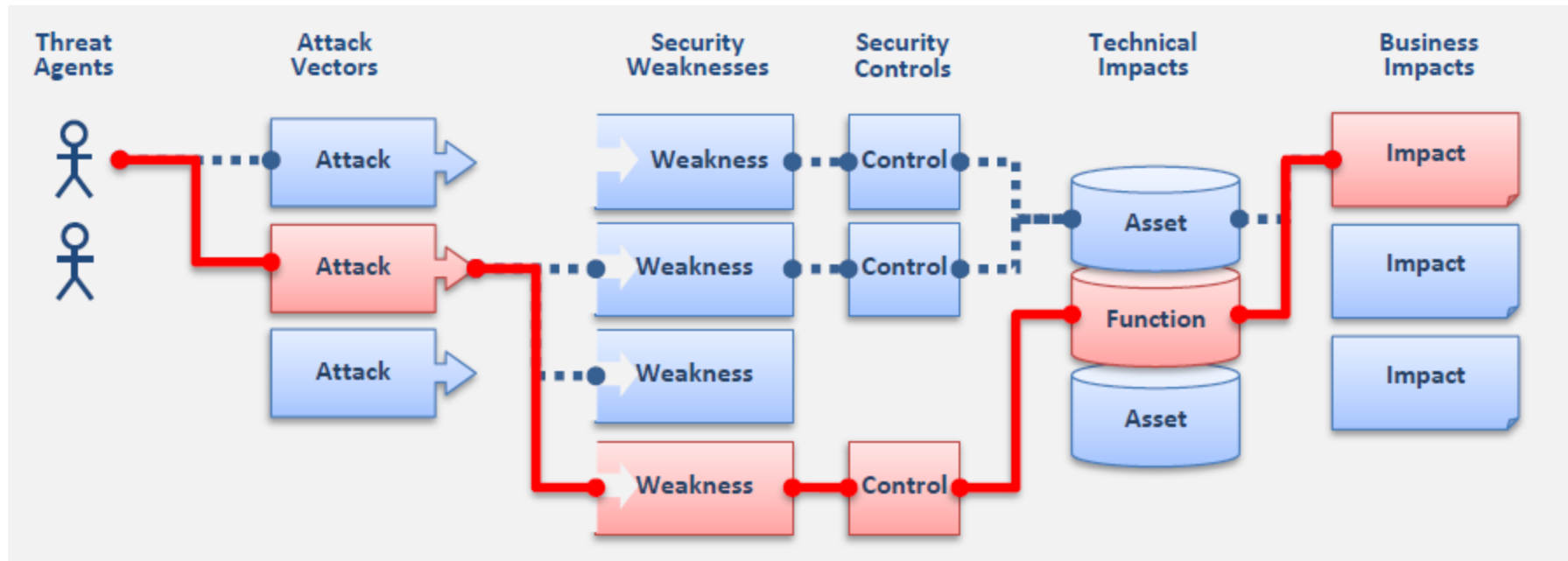


- Testing Principles
- Testing Process
- Custom Web Applications
  - Black Box Testing
  - Grey Box Testing
- Risk and Reporting
- Appendix: Testing Tools
- Appendix: Fuzz Vectors

- Information Gathering
- Business Logic Testing
- Authentication Testing
- Session Management Testing
- Data Validation Testing
- Denial of Service Testing
- Web Services Testing
- Ajax Testing

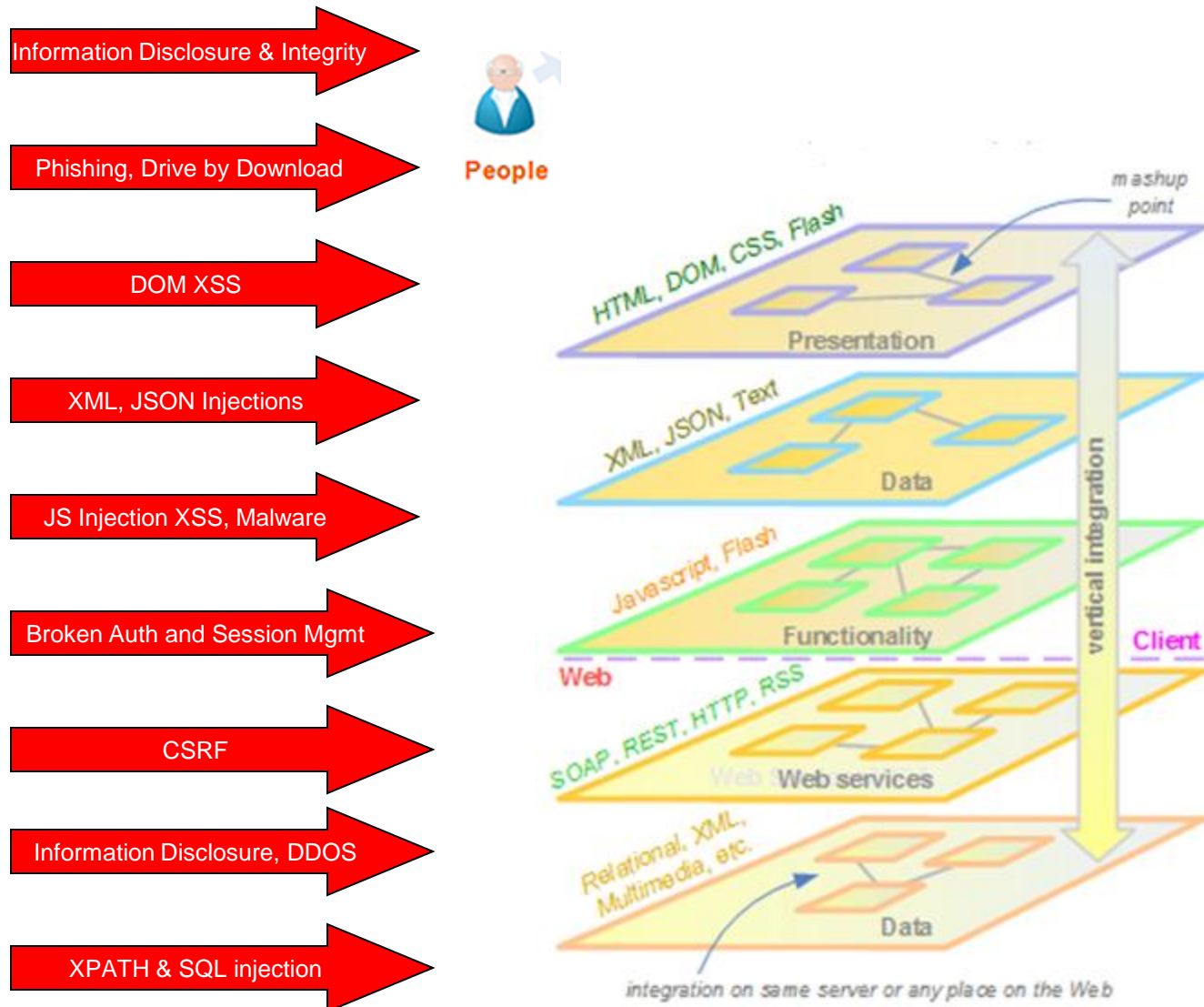
# Web 2.0 Risk Management

# OWASP Risk Framework (used in OWASP T10)



Threat Agent	Attack Vector	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact
?	Easy	Widespread	Easy	Severe	?
	Average	Common	Average	Moderate	
	Difficult	Uncommon	Difficult	Minor	

# Potential Web. 2.0 Attack Vectors And Targets




# Web 2.0 Application Risk Framework

Threat Agents	Misuses and Attack Vectors	Security Weaknesses	Security Controls/ Countermeasures	Technical Impacts	Business Impacts
Web 2.0 Users, Customers/ Employees	User shares private/confidential information, agents post confidential information	Inherent weaknesses in controlling user contributed content in social networks, blogs, IMs, private emails	Web 2.0 Social Networking Security Policies, Compliance, Monitoring, filtering, archiving, approval workflow for social site posts	Loss of sensitive/ confidential data	Reputation loss. Unlawful compliance fines
Malicious Users, Fraudsters	Victim is targeted by phishing, download of phony widgets, clicking on malicious POSTS	Social Engineering, Web 2.0 Vulnerabilities: XSS	Consumer Education, Data Filtering, escape all un-trusted data based on HTML content	Execute JS on client, install malware	Fraud, financial losses, reputation loss/defacements
Malicious Users, Fraudsters	Attacker sends malicious data to the application's interfaces	Web 2.0 Input Validation Vulnerabilities: XPATH injection, XML injection, JSON injection	Filtering, parameterized API, ESAPI filtering APIs, white-list validations	Loss of data, data alteration, denial of service/access	Public disclosure of XSS- Reputation damage
Malicious Users, Fraudsters	Attacker uses leaks or flaws in the authentication or session management functions	Web 2.0 Broken Auth and Session Mgmt Vulnerabilities	Follow Security Requirements For Secure Password Policies, Implement Locking, Disable "Auto-logons"	Unauthorized access to data, functions	Loss of CIA, legal and financial implications
Fraudsters	Attacker creates forged HTTP requests and tricks a victim into submitting them	We 2.0 Cross Site Request Forgery Vulnerabilities	Include the unique token in a hidden field.	Can change data and functions on behalf of the user	Loss of CIA, fraud, denial of access
Automated Scripts/ Spam Bots	Application post links, create accounts, game the application	Insufficient Anti-Automation	Include CAPTCHA, ESAPI intrusion detection APIs	Can overflow process with spam, Enumerations	Business Disruptions/losses, reputational damage

# Web 2.0 Business App Example: Twitter


- Company's Customer Support offers help through twitter's help account, Bank Of America Example





## Bank of America is live in social media

We're here to stay in touch and serve you better.



Twitter Stats

2408	5523
following	followers
Tweets	4965

**You're really communicating with Bank of America. Really.**


Twitter users are known by their "@username" identity. Ours is @BofA\_Help, and the official Bank of America Twitter profile is [http://twitter.com/BofA\\_Help](http://twitter.com/BofA_Help). Contact us here with any questions or concerns.



If you send a tweet to @BofA\_Help (or receive one), you'll be communicating with one of our employees. We know people are talking too, so if we see tweets regarding questions about Bank of America products or services, we'll reach out to see how we can help. We really want to make banking better.

[Go to Twitter](#)


Want to see an example?

Here's how it works:  
You tweet us with a question or concern (or we see a tweet regarding Bank of America and reach out to help). [Read More](#)






Login Join Twitter!




**BofA\_Help**

Welcome to the official Bank of America Twitter site!


Meet our team:



Tara (@tb)



Kasey (@td)



Georgiana (@gs)


[Bank of America](#)

**Important Notice:**  
Never disclose any financial information on Twitter. Bank of America Social Security number, passwords or PINs via Twitter are not responsible for the privacy or practices of Twitter, a Twitter's policies.

## Hey there! BofA\_Help is using Twitter.

Twitter is a free service that lets you keep in touch with people through the exchange of quick, frequent answers to one simple question: What's happening? [Join today](#) to start receiving **BofA\_Help's** tweets.

[Already using Twitter from your phone? Click here.](#)



## BofA\_Help

**@LoveMySkip** I apologize for your experience & would like the opportunity to see if we can help. Plz DM name/zip/phone; no acct numbers. ^tb

4 minutes ago from API in reply to LoveMySkip

[@brookenburris](#) Please DM name, zip, phone and details if there's


**Name** Bank of America  
**Web** <http://social.bankofamerica.com>  
**Bio** We are official Bank of America Twitter reps, here to help, listen & learn from our customers. Tweet with Bank of America reps 8-8 ET Mon-Fri and 9-1 ET on Sat.

2,417	5,533	191
following	followers	listed

**Tweets** 4,984

**Favorites**

**Following**





# Managing Risks of Company's Twitter

## ■ Twitter Application Security Vulnerabilities

### ‣ Landing page for selecting twitter might be vulnerable to web 2.0 vulnerabilities

- **Countermeasure:** Require a scan of web 2.0 vulnerabilities of the landing page hosting the link to twitter

### ‣ Use of AJAX might introduce new source code vulnerabilities

- **Countermeasure:** Validate existence of filtering for sanitization of malicious characters for XSS, XPATH, XML injection and mitigation of CSRF, sufficient anti-automation controls
- **Countermeasure:** Validate compliance of source code with AJAX secure coding standards

# Managing Risks of Company's Twitter

## ■ Twitter Information Security And Compliance Risks

- ▶ **Customers can disclose confidential information by micro blogging to twitter's company account**
  - **Countermeasure** : Ask the user not to enter anything sensitive such as PII, SSN ACC# but his phone number
  
- ▶ **Company is not liable for user's content posted to third party twitter and for twitter vulnerabilities**
  - **Countermeasure** : Once the customer selects to go to twitter he will be presented a speed bump with notice of release of liability to user and to twitter
  
- ▶ **Content shared between enterprise customer support representatives (twitter agents) can leak customer's confidential information such as PII, ACC#**
  - **Countermeasure** : use a content enterprise social filtering and monitoring tool, agents moderate the content that is posted on twitter

# Q & A

QUESTIONS  
ANSWERS

# Thanks for listening, further references

## ■ Ajax and Other "Rich" Interface Technologies

- ▶ [http://www.owasp.org/index.php/Ajax and Other %22Rich%22 Interface Technologies](http://www.owasp.org/index.php/Ajax_and_Other_%22Rich%22_Interface_Technologies)

## ■ Vulnerability Scanners for Flash Components

- ▶ [http://www.owasp.org/index.php/Category:OWASP Flash Security Project](http://www.owasp.org/index.php/Category:OWASP_Flash_Security_Project)

## ■ Web Application Vulnerability Scanners

- ▶ [http://samate.nist.gov/index.php/Web Application Vulnerability Scanners.html](http://samate.nist.gov/index.php/Web_Application_Vulnerability_Scanners.html)

## ■ Facebook Outs Hacker Krillos

- ▶ [http://threatpost.com/en\\_us/blogs/facebook-outs-hacker-kirillos-051310?utm\\_source=Recent+Articles&utm\\_medium=Left+Sidebar+Topics&utm\\_campaign=Web+Application+Security](http://threatpost.com/en_us/blogs/facebook-outs-hacker-kirillos-051310?utm_source=Recent+Articles&utm_medium=Left+Sidebar+Topics&utm_campaign=Web+Application+Security)

# Further references con't

## ■ Facebook Now Trending As Phishing Target

- ▶ [http://threatpost.com/en\\_us/blogs/facebook-now-trending-phishing-target-051310?utm\\_source=Recent+Articles&utm\\_medium=Left+Sidebar+Topics&utm\\_campaign=Web+Application+Security](http://threatpost.com/en_us/blogs/facebook-now-trending-phishing-target-051310?utm_source=Recent+Articles&utm_medium=Left+Sidebar+Topics&utm_campaign=Web+Application+Security)

## ■ Botnet Herders Can Command Via Twitter

- ▶ [http://threatpost.com/en\\_us/blogs/botnet-herders-can-now-command-twitter-051310?utm\\_source=Recent+Articles&utm\\_medium=Left+Sidebar+Topics&utm\\_campaign=Web+Application+Security](http://threatpost.com/en_us/blogs/botnet-herders-can-now-command-twitter-051310?utm_source=Recent+Articles&utm_medium=Left+Sidebar+Topics&utm_campaign=Web+Application+Security)

## ■ OWASP TOP 10 Risks

- ▶ [http://www.owasp.org/index.php/Category:OWASP\\_Top\\_Ten\\_Project](http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project)

## ■ Guide to Twitter Compliance

- ▶ <http://insights.socialware.com/>

# Further references con't

## ■ Web 2.0 Top 10 Web 2.0 Attack Vectors

- ▶ <http://www.net-security.org/article.php?id=949&p=4>

## ■ Defending against the worst web based application vulnerabilities of 2010

- ▶ <http://www.slideshare.net/shreeraj/web-attacks-top-threats-2010>

## ■ Security Concerns Hinder Adoption of Web 2.0 and Social Networking in Business

- ▶ <http://investor.mcafee.com/releasedetail.cfm?ReleaseID=511103>

## ■ Web 2.0 a Top Security Threat in 2010, Survey Finds

- ▶ <http://pr.webroot.com/threat-research/ent/web-2-security-survey-170210.html>