yankee group report

October 2007

by Andrew Jaquith, Enabling Technologies Enterprise, Compliance, Security and Risk Management Program Manager,
ajaquith@yankeegroup.com, 617-598-7351

The Web 2.0 Security Train Wreck

The Bottom Line:	Web 2.0 technologies offer much promise for enterprises looking to build rich internet applications, but underlying security issues are serious and unlikely to be solved soon. Enterprises riding the Web 2.0 train should expect a rough ride ahead.
Key Concepts:	Web 2.0, web applications, security
Who Should Read:	CIO, CSO

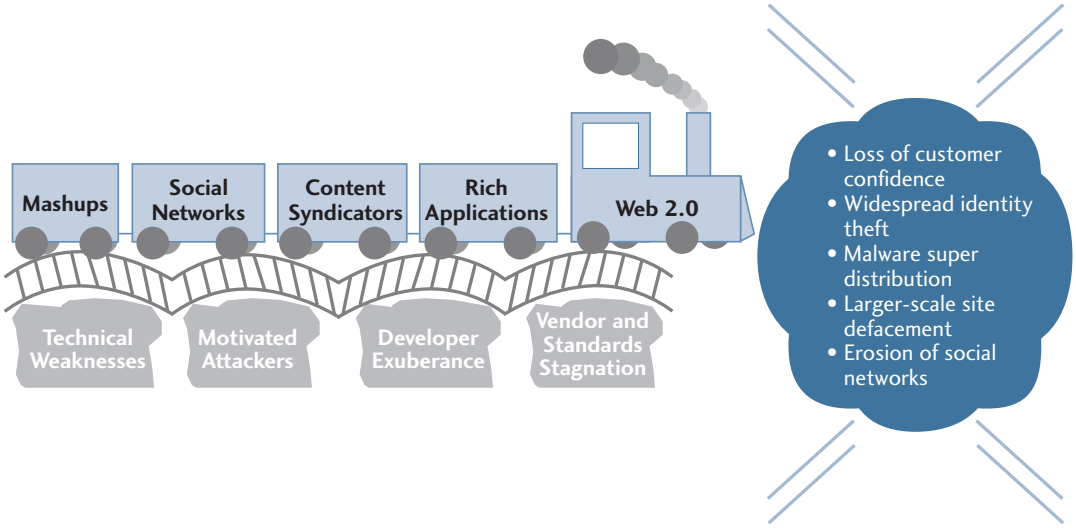
Practice Leader: Zeus Kerravala, Enterprise Research Senior Vice President, zkerravala@yankeegroup.com, 617-598-7235

Executive Summary

Web 2.0 technologies offer much promise for enterprises looking to build rich internet applications, but underlying security issues are serious and unlikely to be solved soon. Web 2.0 technologies exacerbate old methods of attack, and enable new ones. Exploitation of Web 2.0 applications will lead to:

- Widespread identity theft and transaction fraud
- Malware super-distribution
- Large-scale web site defacement
- Erosion of social networking memberships
- Loss of consumer confidence

Exhibit 1.
Web 2.0's Security Problems Causing a Slow-Motion Train Wreck
Source: Yankee Group, 2007



To fix the Web 2.0 security problem, browser vendors should join online destination web sites and selected security researchers to strong-arm the W3C into action, kill off dangerous HTML features, fix JavaScript, and define a browser security model that accommodates mashups.

Enterprises riding the Web 2.0 train should expect a rough ride ahead (see Exhibit 1). In the absence of immediate fixes, they should:

- Scrub custom web application code
- Select third-party applications and toolkits with care
- Avoid PHP-based collaborative applications
- Assess existing public-facing applications for security vulnerabilities
- Shield vulnerable public-facing applications
- Block user traffic going to dangerous web sites
- Delay deployment of high-value, customized Web 2.0 applications

Table of Contents

I. **Introduction** **3**

II. **Web 2.0 Changes the Application Landscape** **4**

The Technologies Underpinning Web 2.0 4

III. **Web 2.0's Creaky Security Foundations Erode** **7**

Best Practices for Traditional Web Applications Are Not Always Followed, but They Are Well Understood 7

Web 2.0 Applications Ignore Lessons Learned the Hard Way 8

Cross-Site Request Forgery 9

Malware Injection into RSS Feeds 10

JavaScript Hijacking 10

Forum Poisoning 10

IV. **Web 2.0 Security Predictions** **11**

Vendor Response 12

V. **Conclusions and Recommendations** **14**

Recommendations for Browser Vendors and Online Services 15

Recommendations for Enterprises 16

VI. **Further Reading** **17**

I. Introduction

Web 2.0 technologies offer much promise for enterprises looking to build rich internet applications, but underlying security issues are serious and unlikely to be solved soon. Enterprises should proceed with caution, hire web application security experts and take steps to keep their customers safe.

In this Yankee Group Report, we describe:

- How the Web 2.0 application wave is changing the way enterprises reach customers
- What technologies underpin modern Web 2.0 applications
- How weaknesses in the dominant Web 2.0 application technologies will lead to new classes of attacks on applications and business models
- How security vendors are responding to the emerging Web 2.0 application security problem
- What steps enterprises should take to protect their application portfolios

II. Web 2.0 Changes the Application Landscape

Across the technology landscape, the Web 2.0 application revolution has arrived. The combination of open programming platforms, cross-platform/cross-browser communications protocols and generous dollops of venture capital has produced an explosion in the number of Web 2.0 applications available to end users. Originally coined by Tim O'Reilly, the term "Web 2.0" is generally taken to mean applications that include:

- **Rich application interfaces** that provide immediate feedback to users based on data requests that are processed asynchronously "in the background," and without waiting for a full HTTP page refresh. Modern rich Web 2.0 applications rely heavily on the combination of server-side interfaces and corresponding client-side JavaScript, XMLHttpRequest and JavaScript Object Notation (JSON). These features allow developers to build applications that look and act like desktop applications. Google's Maps and Gmail applications are well-known examples of snappy, responsive interfaces.
- **Content syndication features** that stream new web site content, such as blog posts or site changes, into data feeds that client applications consume. Syndication formats include Rich Site Summary (RSS) and Atom. Because RSS feed-reading cannibalizes ordinary site traffic, data feeds increasingly include content sourced from third parties such as advertisers.
- **Social networking sites** that encourage contributions from end users, and whose value depends on continuing participation. These kinds of sites include wikis (editable web sites), blogs, presence/instant messaging sites such as Twitter, "bulletin board" community interest sites, and networking sites such as YouTube, Facebook and MySpace. Social networking sites follow Metcalfe's Law: The value of the network is a function of the square of the number of users in it.
- **Cross-domain mashups** that combine functions from multiple web domains into a single application. Most of the earliest mashups combine mapping or geolocation features with other data. For example, Trulia.com combines real estate listings and Google Maps data into a dynamic display that charts real estate prices and historical sales data together. Eyebeam's Fundrace.org web site allows visitors to discover and geocode US Federal Election Commission political contribution data for particular street addresses and zip codes.

Although the specific objectives and audiences of these four styles of Web 2.0 applications are different, they all share one trait in common: using open, industry-standard protocols to capture, captivate and involve end users.

The Technologies Underpinning Web 2.0

Web 2.0 sites rely on a grab bag of internet technologies to do their magic (see Exhibit 2). Sites with syndication features, for example, typically use server-side script code (e.g., PHP, Perl, JSP) to generate on-demand and cached RSS and Atom feeds that modern browsers (Safari, Internet Explorer 7, Firefox 2) and dedicated RSS readers (NetNewsWire, FeedDemon, NewsGator) know how to parse and display.

Other kinds of Web 2.0 applications (rich applications, mashups, social networking) mix client-side and server-side code components. Common functions include:

- **Back-channel data exchange protocols** that invisibly request, fetch and process data from remote web servers and refresh parts of the current HTML page, such as a map window or drop-down list. The most popular techniques use the JavaScript XMLHttpRequest function ("AJAX") or standard HTTP GET operations to return chunks of XML or JavaScript Object Notation (JSON), respectively. Back-channel data exchange is the workhorse of modern Web 2.0 applications, and is what gives them their responsiveness and familiar "desktop-like" feeling. Google Maps was arguably the first significant Web 2.0 application to aggressively use back-channel data exchange.
- **JavaScript toolkit frameworks** that automate the process of building internet applications that have more dynamic effects and communications capabilities than plain old static HTML. These toolkits package up common JavaScript functions to render user interface widgets such as calendar date pickers, dynamic drop-down menus and rollover effects. They also typically automate the process of setting up back-channel JSON or XML connections. Popular client JavaScript toolkits include the Dojo Foundation's eponymous toolkit, Prototype and Google's GWT framework.
- **Server-side scripting languages** that process user requests and return data to the browser (or to the back-channel caller invoked via JavaScript). Back in the days when web applications didn't contain much dynamic client code ("Web 1.0"), nearly all web applications were built strictly with server-side code. Traditional web scripting languages used on servers include the various Microsoft .NET languages (ASP.NET), Perl, PHP, and JSP (Java Server Pages). A newer framework, Ruby on Rails (RoR) has lately been making much headway because of its simplicity and easy integration with client-side JavaScript.
- **Server-side widget frameworks** and **client-side inclusion libraries** that allow developers to create mashups that freely mix functionality across domains, but appear to the user as a single application. Applications are mashed up either on the server side using frameworks such as Netvibes or PageFlakes, or on the client via JavaScript libraries like Dojo.

Significantly, most of the underlying technologies can be used on all operating systems, and with all browsers. Many of the client toolkits and JavaScript libraries are also open source or dual-licensed, making them very attractive to startups and independent developers. Ubiquity, developer involvement and low licensing hurdles are keys reason why the Web 2.0 application scene is so vibrant.

The most significant consequence of Web 2.0 technologies has been to kill off legacy toolsets tied to specific platforms. Although back-end servers continue to require vendor-specific interfaces and languages, few companies develop large-scale *client-side* applications this way any more. Web technologies such as HTML, cascading style sheets (CSS), Flash and JavaScript are winning the mind share war against legacy Win32 toolkits. Today, the number of job postings for JavaScript already exceeds the demand for Microsoft's Visual Basic and Java Swing combined (source: Indeed.com).

With the advent of rich client features such as JavaScript and back-channel data exchange protocols, Web 2.0 represents the fourth major shift in programming styles: from green-screen (server only) to desktop PCs (client/server); to HTML and web-based applications (predominantly server-side) to Web 2.0 (mixed client and server code). The last major mixed-mode programming wave (client/server) assumed private networks without exposure to outside actors. But Web 2.0 applications make no such assumptions; by definition, the networks are public. This has interesting—and obvious—implications for security, which we describe next.

Exhibit 2.
Web 2.0 Technologies

Source: Yankee Group, 2007

Feature	Sample Websites	Enabling technologies
Rich Application Interfaces	• Google Gmail • Google Maps • LinkedIn • Wachovia Online Banking	AJAX (Asynchronous JavaScript and XML) Back-channel data exchange protocols: • XMLHttpRequest • JSON JavaScript toolkit frameworks: • Google GWT • Dojo Toolkit • Prototype • Microsoft Atlas Client-side data storage: • Google Gears
Content Syndication	Individual sites with RSS: • The New York Times • Boing Boing • Techcrunch • Gartner Watch Syndication aggregators:· • FeedBurner • Technorati	Rich Site Summary (RSS) ATOM
Social Networking	• Twitter • Flickr • Facebook • Wikipedia • iLike • AOL Mgnet	Blog/wiki software: • MediaWiki • JSPWiki • WordPress Internet forum software: • PHPBB • Invision Power Board • vBulletin • JForum Sever-side scripting languages • PHP • JSP • ASP/ASP.NET • Ruby on Rails
Cross-Domain Mashups	• Trulia (Google Maps + real-estate listings) • Velyoo (Google Maps + eBay + Amazon) • Salesforce.com + Google Maps • Fundrace.org (FEC political contributions data + Google Maps)	Server-side widget frameworks: • Netvibes • PageFlakes • J2EE portlets (JSR-168 and JSR-286) • Salesforce.com AppExchange Client-side inclusion: • Dojo Toolkit • Custom JavaScript

III. Web 2.0's Creaky Security Foundations Erode

Web 2.0 applications make the internet more interesting, inviting and integral to the lives of consumers and enterprise employees. However, Web 2.0-style applications have potentially more serious security exposures than traditional web applications ("Web 1.0").

Best Practices for Traditional Web Applications Are Not Always Followed, but They Are Well Understood

As background, consider the security needs of server-centric Web 1.0 applications. Best practices for developing secure applications, while not universally followed by all developers, are at least reasonably well understood. The most important principles, as advocated by the Open Web Application Security Project (OWASP) and others, include these:

- **Use HTTPS to protect sensitive data** in transmission, such as user names and passwords.
- **Do not trust the user.** Applications should not assume anything regarding the current state of the user's browser. Session state should be stored server-side rather than stored in an untrustworthy browser, such as in cookies.
- **Validate all user-submitted data.** Applications should validate all user-submitted form parameters, and should screen out potentially hostile input such as tick marks ('), angle brackets and special characters. Good input validation prevents attacks such as SQL Injection, which has been used (infamously, in several cases) to cause applications to disgorge large volumes of sensitive data.
- **Sanitize all generated output** to prevent the chance that user-submitted content from being rendered as hostile JavaScript or HTML. In extreme cases, failure to sanitize output enables a type of attack called cross-site scripting (XSS), which in some cases has caused legions of web sites (such those that use the PHPBB web site) to become massively and serially infected with injected malware.
- **Write server-side code using typesafe languages**, such as Java/JSP and ASP.NET, to prevent buffer overflow attacks that could cause the application stop functioning.
- **Expire idle user sessions quickly** to reduce the chance that an outsider who steals a user's session can cause damage to the account.
- **Require additional authorization for high-value transactions** such as goods purchases, stock trades and changes to sensitive account information. Authorization methods may include requiring the user to supply their account password, confirm the transaction via an out-of-band channel such as SMS, or enter credentials supplied by a one-time password token or smart card.

Not all application developers (or the enterprises they work for) follow secure web programming principles all the time. However, vertical market segments with traditions of strong security programs, such as investment banking and commercial banking, have placed special priority on auditing web applications before they go live, to ensure that they adhere to the safe programming practices described previously.

Secure web development principles have, in turn, trickled down into contractual standards such as the Payment Card Industry's Data Security Standard. Version 1.1 of PCI-DSS (September 2006), for example, requires that firms who accept Visa, MasterCard, Discover, American Express, or JCB credit cards for transactions must perform annual web penetration tests or code reviews on their applications.

Web 2.0 Applications Ignore Lessons Learned the Hard Way

Many security researchers argue that Web 2.0 applications are no different than are web applications after all; they inherit the same security issues as the traditional kind. Jeremiah Grossman, the CTO of WhiteHat Security, argues: "AJAX does not make a web site more insecure in and of itself." However, some of the technologies commonly used in Web 2.0 applications make security vulnerabilities easier to exploit, and in many cases open up entirely new classes of attack because of:

- **Pervasive use of JavaScript.** Most modern Web 2.0 applications rely heavily on JavaScript. But its security weaknesses, such the ability of loaded code to redefine language functions, are manifold and well known. Worse, the current custodian of the JavaScript standard, the ECMA, has little clout with developers and browser makers to fix features prone to abuse.
- **Excessive trust of client-side code.** The "back to the future" shift toward client-side web programming has caused programmers to forget that the wire connecting clients and servers (the internet) is untrusted. Many Web 2.0 server applications make improper assumptions about the state the client should be in, and fail to validate user-supplied input correctly.
- **Implicit trust of foreign code domains.** Mashup applications and advertising insertions load foreign HTML and JavaScript, whose functions and security capabilities are unknown. Successful infiltration of a script sourced from a foreign site could cause a complete compromise of user sessions and disclosure of confidential information.
- **Lack of browser security model.** Although some browsers are better than others, no industry consensus exists on how to define security policies that work on all browsers. Current restrictions did not envision Web 2.0-style applications. Those conventions that exist, such as the so-called "same origin" policy for XMLHttpRequest objects (used for making back-channel data requests), are only a small subset of what is needed. Alex Russell, the project lead for the Dojo Foundation, says "browser manufacturers have lost their forum for agreeing on things… The W3C hasn't done anything useful for the last 5 years."
- **Exuberant developers pushing security boundaries**. To make it easier to create mashups, Web 2.0 toolkit developers are actively developing techniques to bypass what browser security restrictions still remain, such as the same-origin policy. According to Dojo's Russell: "We are bending the rules… and we can blithely say that we don't do more than what the browsers will allow."
- **Rookie developer mistakes.** Despite the best efforts of organizations such as OWASP, SANS and WASC (Web Application Security Consortium), developers bent on adding features at breakneck speeds continue to make the same security mistakes over and over—lack of input validation, unsafe assumptions about deployment environments, configuration mistakes and generally poor security design. Several popular applications based on the PHP framework, such as phpBB, have appalling security track records. This is partly because of deficiencies in PHP itself, but also because PHP developers tend to be "scripters" without formal security training.

These issues—placing excessive trust in pervasive, client-side JavaScript, lack of proper browser security controls, and developer exuberance—have ushered in new techniques for injecting malware and stealing information. New attack techniques include:

- Cross-site request forgery (CRSF)
- Malware injection into RSS feeds
- JavaScript hijacking
- Forum poisoning

We explain these techniques, and their consequences to enterprises, next.

Exhibit 3.

Web 2.0 Security Weaknesses

Source: Yankee Group, 2007

Feature	Security Weakness
Rich Application Interfaces	Cross-site request forgery (CRSF) Session stealing
Content Syndication	Cross-site scripting (XSS) Malware injection
Social Networking	Cross-site scripting (XSS) Malware injection Forum poisoning Mass defacement Spam generation
Cross-Domain Mashups	Cross-site scripting (XSS) Cross-site request forgery (CRSF) Session stealing Malware injection

Cross-Site Request Forgery

Cross-site request forgery (CRSF) is a technique used to, via a malicious JavaScript function, cause a user to automatically initiate transactions in the background without his or her knowledge. CRSF takes advantage of users who are currently logged in to financial or commercial web sites. If a malicious script detects that a user is logged in to targeted web sites, it can buy or sell stock, move money to offshore bank accounts or divert sensitive information on his or her behalf.

CRSF attacks are aided significantly by the presence of cross-site scripting (XSS) vulnerabilities on web sites. Enterprises whose web sites fail to correctly filter user input, and fail to properly sanitize output when it is re-displayed, are at risk of inadvertently hosting hostile JavaScript files that could be used to mount attacks. As such, correct validation and sanitization is now a higher priority than ever before.

Security consultants Yankee Group spoke with agree that CRSF is the sleeper security issue of modern web applications. Grossman of WhiteHat Security says: "There are 100 million web sites out there. Of the 100 or more that we assess every month, 80% of them are vulnerable to CRSF." ISEC Partners' Alex Stamos echoes this view: "CRSF is one of our favorite attacks. Just about every one of our customers is vulnerable to it."

Malware Injection into RSS Feeds

Several security vendors, such as ScanSafe, Authentium, Trend Micro and SPI Dynamics, have predicted that RSS and Atom feeds will become an increasingly important threat vector for malware, and will supplement the most popular ones (e-mail and web browsing). Under this scenario, attackers compromise host web servers and inject malware or harmful JavaScript into existing RSS feeds. The injected malware then propagates across the internet to all subscribers of the feed. Note that "injection into an existing feed" need not require an attacker to actually compromise a site. Injection might be as simple as adding comments to blog pages—which in many cases are streamed out to subscribers as part of the RSS feed.

RSS malware injection is particularly worrying for several reasons. First, the robustness and resiliency of RSS reader code is significantly less well understood than that of the most popular e-mail clients and web browsers. These applications have had a significantly longer amount of exposure to buffer overflows and hostile scripts. Second, the RSS readers themselves do not generally have any built-in extension APIs that would allow standard anti-malware software to be installed. This is different than browsers and e-mail clients, which increasingly have features such as anti-phishing filters to spam-trapping algorithms.

As a result, RSS readers have few natural defenses against malware injected into feeds they parse, leaving enterprises totally reliant on gateway-based defenses. Although the risk today is relatively low, the potential for abuse will increase as client operating systems such as Windows Vista and OS X integrate more RSS features into the OS and browser.

JavaScript Hijacking

Publicized by Fortify Software in March 2007, and building on research by WhiteHat's Grossman on a series of vulnerabilities in Google's Gmail application, JavaScript hijacking exploits the fact that the JavaScript language allows language elements, such as object constructors, to be redefined. This feature is especially useful for applications that require the ability to communicate with third-party sites in the background, such as mashups.

However, in the hands of an attacker this feature can be used maliciously. If an attacker can coerce a user into visiting a page that returns specially coded JSON messages to the browser, the messages will be parsed and executed as live JavaScript code. Practically speaking, this allows an attackers' code to observe and redirect page contents and confidential data to a third-party web site of the attackers' choosing.

JavaScript hijacking works because it takes advantage of a developer "feature" whose consequences were not envisioned back in 1995, when Netscape invented JavaScript. Application functionality—and its evil-twin security failure modes—have evolved significantly since then, with the familiar result being that developers must choose between flashy features and security. As Fortify's Brian Chess puts it, "An application can be mashup-friendly or it can be secure, but it cannot be both."

Forum Poisoning

Collaborative Web 2.0 applications often include a significant number of features designed to make it easy for users to register and participate in online forums, blogs and group activities. Modern, web-enabled bulletin-board software such as phpBB and JForum have been adopted by corporations looking to speed internal projects and connect with their customers. Wiki software accomplishes a similar objective, using the editable web page as the primary method of participation.

Making it easy for customers to self-register lowers barriers to participation. Enterprise users of open source forum software, for example, include Jenny Craig and *The New York Times*. However, the lowered participation barrier extends to malicious actors, not just legitimate users. Today, criminals can use automated tools such as Webattacker and XRumer to create fake users and post massive numbers of spam messages containing links to malware sites.

Forum poisoning techniques become more effective when combined with cross-site request and other coding vulnerabilities inherent to many Web 2.0 collaboration applications. For example, in May 2006 a malicious party exploited a weakness in a version of Invision Power Board hosted by Circuit City's customer support organization. The attacker planted a message containing the infamous Microsoft WMF image exploit, which as many as 100 people viewed. Those who did not have up-to-date security patches were compromised instantly.

IV. Web 2.0 Security Predictions

Yankee Group believes that the security issues associated with Web 2.0 applications are serious enough to warrant raising the alarm. The browser security model is broken, and endemic programming weaknesses increase the likelihood of attacks. Watchfire's CTO, Danny Kim, summarizes the state of Web 2.0 security today: "The root of the problem is that you are pushing parts of an application to people you can't trust. The mantra about web app security has always been that you can't trust the client. Now that half of the application is being pushed out to the client, the attack surface has fragmented. Now you can't even tell where things are coming from."

In the coming 12 to 18 months, we believe that malicious parties—who largely write malware targeting Windows operating systems—will practice their dark arts on the web instead. Future malware conjured up by online criminal gangs and contracted, full-time malware writers will be much more platform neutral. Sana Security's Vlad Gorelik expects that "malware will target the browser through JavaScript and other active content. We are going to see more and more man-in-the-middle and session riding attacks targeted at the browser itself; these are likely to be written in JavaScript." Exploitation of Web 2.0 applications will lead to:

- **Widespread identity theft and transaction fraud:** On Windows platforms, the spyware problem has affected financial institutions' bottom lines. In 2006, for example, E-Trade and TD Ameritrade wrote down $35 million in losses because of customer identity theft. These numbers will increase dramatically because of an explosion in collaborative Web 2.0 sites that serve up Trojan horses designed to steal corporate and consumer identities. The Circuit City example is just the beginning of what we believe will be a long-term trend.
- **Malware super-distribution:** Attackers will increasingly exploit programming weaknesses and used to implant traditional OS-specific malware. In June 2007, Finjan discovered a popular video sharing site called HotGet.com (Alexa rank: top 2000) that had been compromised; its web pages contained JavaScript that attempted to download a nasty spyware program, MPack, to users' machines. According to Finjan's Ben-Itzak, "the main site was infected; every piece of video users try to share was also infected." Infiltration of social media sites is now a key target for criminals, according to Finjan: "Web 2.0 is good for you and me as honest people. But it is also great for the hackers."

- **Large-scale web site defacement:** The increased usage of common Web 2.0 platforms for collaboration means that exploitable programming weaknesses will be used increasingly for large-scale defacements. For example, in May 2005 a Turkish hacker exploited a weakness in a common GoDaddy web page (gdform.asp) and defaced 38,000 web sites in a single day. Although the hack was done to make a political point and designed to attract attention, Yankee Group expects to see many more incidents that are much less obvious. Future attacks on common application infrastructure will be considerably more subtle, subversive and security imperiling.
- **Erosion of social networking memberships:** Web sites that lose their reputations for safety will lose members in droves, not to mention advertising revenue. Finjan's Ben-Itzak believes that popular social networking sites are serious risk of forum poisoning. "We used to see 10 infected MySpace pages a day. It's gotten so bad that we've stopped reporting them." Free blogging sites that have limited defenses against automated registration tools, are particularly at risk. Security vendors estimate that as many as one-third of Blogger and Blogspot accounts contain links to malware sites. Failure to safeguard the user experience will cause memberships in vulnerable web sites to plummet.

Vendor Response

The market for security tools and technologies that address Web 2.0 security issues is in flux. Web 2.0 shares many traits with traditional, server-centric Web 1.0 sites. As such, products that deal specifically with Web 2.0 security concerns are often seen as brand extension opportunities for incumbent vendors. Security vendors are currently selling (or will soon sell) software and services to address Web 2.0 security threats, which include:

- **Web site security scanning vendors:** Vendors who specialize in probing web applications for vulnerabilities (a $70 million market) are already making strong moves into the Web 2.0 security space by adding JavaScript scanning features. These vendors include Cenzic, SPI Dynamics (recently acquired by Hewlett-Packard), Watchfire (recently acquired by IBM) and Core Security. However, to be successful in the Web 2.0 world, black-box fuzzing and scanning won't be enough; "white-box" language runtime simulation and flow analysis from code assurance vendors will be needed. We expect M&A activity in 2008 and 2009 from firms looking to build more complete application scanning portfolios.
- **Application vulnerability management services:** SaaS vendors such as WhiteHat Security offer many of the same features as standalone web site security scanning products, but speed time to value via managed services. WhiteHat, for example, creates, hosts and executes customer-specific application tests that combine tool-assisted remote assessment with ad-hoc consulting. WhiteHat's subscription model offers an intriguing, cheaper and more scalable alternative to the classic "two people for 2 weeks" consulting. Over time, we expect small-to-midsized firms like WhiteHat to increasingly mechanize assessment services and spin out dedicated Web 2.0 products, similar to the way McAfee's Foundstone unit did in the first half of this decade. Driven by the PCI certification boom, the hosted anti-virus managed service (AVMS) and web site security scanning segments will merge as SPI Dynamics, WatchFire and others increase their SaaS efforts (e.g., WebInspect Direct, AppScan On Demand). Yankee Group also expects VMS vendors that focus more on operating system vulnerability scanning, such as Qualys, to move into AVMS as well.
- **Code assurance vendors:** Although the market for products that perform static and dynamic analysis on application code is not large, the shift back toward client-side code plays to the strengths of vendors with application knowledge such as Fortify, Klocwork and Ounce Labs. However, these vendors' product menus are predominantly oriented toward server-side languages such as Java, C# and C++. A generous helping of JavaScript, and a dash of Flash, would make their pitches go down much easier with enterprise customers embarking on Web 2.0 initiatives.

- **Web application firewall vendors:** Traditional web application firewall vendors (Imperva, NetContinuum) are well positioned to safeguard enterprise Web 2.0 applications because they can filter user input, sanitize output, forcibly expire user sessions and provide some defenses against cross-site scripting (XSS) and cross-site request forgery (CRSF). Citrix and F5, two large networking companies that acquired web application firewall businesses in 2005 (Teros and the firewall division of WatchFire), have highly diluted messages and are not as focused; nearly 2 years after acquisition, for example, F5 can barely put together a $2 million quarter for its application firewall products. By contrast, we expect specialist stack protectors like Imperva to thrive. We also expect wild cards like Barracuda to enter the market; they will drive large volumes of web app firewall sales at attractive prices.
- **Application security consultants:** Specialized internet application consultants such as Security Innovation, NGS Software, Leviathan Security and iSEC Partners already have deep knowledge in how modern Web 2.0 applications work. They will continue to appeal to customers who want bespoke assessments (black-box and white-box) and remediation guidance.
- **Web filtering products:** Vendors such as WebSense, ScanSafe, Finjan, Secure Computing, and McAfee with its SiteAdvisor product, offer products that block employees from surfing to web sites that harbor dangerous malware. These vendors will attack the Web 2.0 malware super-distribution problem first. However, as malware infiltrates collaborative sites in more widespread ways, site-blocking will become too much of a blunt instrument. Malware posted to the New York Times' reader forums shouldn't cause the entire web site to be blocked. We expect that the next generation of products, arriving in the 2008-2010 time frame, will discriminate at increasingly finer levels of detail, for example, by filtering individual pages or posts.

In addition to these vendors, large diversified companies such as IBM, Symantec and Microsoft have varying portfolios and capabilities to address Web 2.0 security concerns:

- **Microsoft:** Always strong on architectural guidance and prescriptive practices, Microsoft's Web 2.0 application security efforts are deeply embedded in its Secure Development Lifecycle and its related .NET application stack. While we believe it is likely that Microsoft will add more security features to its server-side frameworks (ASP.NET) and web application firewall products (née Whale), over time we expect that it will not participate broadly in the client side other than through efforts to improve its own client-side web toolkit (Atlas). Its value to heterogeneous environments, particularly those using open source collaboration stacks and JavaScript toolkits, is minimal. As a result, in the short to medium term, Microsoft—as a total application security solutions provider—won't provide much value for mixed environments unless they use Microsoft products top to bottom.
- **Symantec:** For all its rhetoric about "protecting interactions," Symantec's capability to secure Web 2.0 applications is approximately zero. Its applications consulting unit (née @stake) has been decimated by defections. Other than its Web Security perimeter web-filtering product, and its anti-phishing client software (which we consider to be out of scope of this Web 2.0-focused report), its application security product portfolio is extremely thin. Today, Symantec is a nonfactor in the Web 2.0 security market. But that could change quickly; Symantec has historically waited for nascent security markets to mature before buying its way into them.
- **IBM:** Unlike Symantec and Microsoft, IBM is well positioned to offer a full-service Web 2.0 application security portfolio. Its recent Watchfire acquisition gives it the market-leading web application security scanner; its Rational properties give it System Development Life Cycle (SDLC) tools. Moreover, IBM has left its Internet Security Systems (ISS) division, which employs consultants who are well versed in web application vulnerabilities, largely intact. Perhaps most important, IBM contributes code, money and developer talent to the Web 2.0 open source community, for example, the Dojo Foundation. That said, these four sources of strength are scattered over four different divisions, and IBM has made no real attempt to unify them into a coherent service platform, let alone a coherent set of messages.

The Web 2.0 security problem is diffused over so many different market categories that it is highly unlikely that a dedicated market for products will emerge, at least in the short term. Yankee Group expects that the likeliest outcomes will be opportunistic expansions into adjacent markets by current participants. We will also see cross-segment mergers, such as between vendors in the code assurance and web application scanning markets.

Finally, in the next 24 months, Yankee Group expects that several vendors will bring client-side products to market that inject rules and behavioral heuristics into browser runtimes, Essentially, this means embedding a miniature host intrusion prevention system (HIPS) into Internet Explorer, Firefox and other browsers. Last year, the Windows kernel was the battleground. Next year, it will move to inside the browser—with all of the chaos, API hackery and vendor brawling that that shift implies.

V. Conclusions and Recommendations

Web 2.0 application security is a slow-motion train wreck. We can already see the sparks on the tracks, and hear the squeal of the brakes in the distance. Predicting that Web 2.0 applications will, collectively, turn into a security disaster is an easy call.

However, like most slow-motion disasters, the damage will only be obvious in retrospect. Yankee Group believes that left unchecked, it will not be until at least 2010 until widespread knowledge of Web 2.0 attack vectors and their corresponding defenses are understood by vendors, mainstream enterprises and open source project teams.

By themselves, Web 2.0 security issues could be solved with concerted effort from key internet stakeholders such as the major browser developers (Microsoft, Mozilla, Apple), large internet portal and search vendors (Yahoo!, Google, Microsoft), toolkit vendors and security researchers. Yankee Group sources tell us, confidentially, that multi-party talks between many of these exact stakeholders fell apart in the latter half of 2006.

With limited prospects of consensus, the current Web 2.0 security situation has become essentially intractable. Web 2.0's primary programming language, JavaScript, has no active champion. No party has the clout or power to fix issues with the language or with related cross-domain applications. Meanwhile, there continues to be widespread deficiencies in the way most applications are built.

In the meantime, the only evidence of trouble will be a slow, entropic loss of consumer confidence, occasionally interrupted by spectacular security breaches at vulnerable Web 2.0 destinations. Symantec CEO John Thompson, at his company's June 2007 Vision conference in Las Vegas, wondered aloud whether internet security issues have already decreased consumers' willingness to buy products online.

Recommendations for Browser Vendors and Online Services

The Web 2.0 security problem has been several years in the making, and it will take a similar amount of time to unmake the mess. The most important browser vendors (Microsoft, Mozilla, Apple, Opera) should join together with the world's largest online destination web sites (Google, eBay, Yahoo!, Amazon) and selected security researchers to define a browser security "Contact Group." This group's charter should be to add better underpinnings to Web 2.0 applications' shaky security foundations, starting with the browser. Namely:

- **Strong-arm the W3C into action, or declare it dead.** During the last 5 years, the World Wide Web Consortium (W3C) has pursued costly fool's errands like the Semantic Web, while the HTML specification suffered from benign neglect. Its efforts would have been better served helping create more secure versions of HTML. Current drafts of HTML 5 with the W3C (as initiated by WHATWG) reflect the sensibilities of web designers rather than security analysts, and are sorely deficient. Instead, the Contact Group should storm the gates of the W3C and compel it into action on securing HTML 5. Member companies should also dedicate security staff to ensure that the job is done right.
- **Kill off dangerous HTML features.** Legacy web features prone to abuse (such as IFRAME, ironically introduced by Microsoft for Internet Explorer 4) are often used by ad vendors, but equally by malicious attackers. These should be deprecated and removed from the HTML specification, or disabled by default.
- **Define a browser security model that accommodates mashups.** Early drafts of HTTP and HTML 5 contain some provisions for compartmentalizing web pages so that foreign domains can communicate with each other safely. Other good ideas include the "HTML modules" proposal by Yahoo!'s Douglas Crockett and Microsoft's "HttpOnly" proposal for safe cookie handling. In addition, the Browser Contact Group should also jointly define a common standard for declarative security policies in browsers. These would provide a sandboxed execution environment for JavaScript and rich internet application technologies such as Flash, Microsoft's Silverlight and Adobe Apollo similar to the way managed JRE and CLR runtimes constrain Java applets and .NET assemblies. These ideas, when combined with innovations such Gervase "Gerv" Markham's Content Restrictions proposal, would close the most dangerous cross-domain security holes while allowing trusted applications to safely define the domain and operating system permissions they need.
- **Engage web toolkit makers to help fix JavaScript.** Browser vendors (notably Microsoft) previously drove innovation in the web application space while developers struggled to keep up; the reverse is now the rule. Web 2.0 developers are chafing against the few browser security restrictions that remain. Lack of action by vendors is making developers cranky. Alex Stamos notes: "[Developers] think security people are evil. When you look at the message boards you see a lot of bitching and moaning about how the security people won't let them do things." Constructive engagement with the leading toolkit makers (Dojo, Prototype, Script.aculo.us), many of which are open source collectives, would do much to reduce animosity and forge agreement on how to move forward.

Unfortunately, these recommendations may not come to pass any time soon. Browser makers have different agendas. Microsoft's interest in fixing JavaScript is low because the language wasn't one it invented, and because of its desire to promote Silverlight, a rich internet client (RIA) language and toolset. Adobe is similarly constrained due to its interests in Flash and the Apollo RIA framework. In the end, Google, Mozilla and Yahoo! will be the most motivated parties.

Recommendations for Enterprises

The rise of Web 2.0 applications has coincided with the emergence of software vulnerabilities that threaten to undermine businesses that depend on the new technologies. But no magic bullet exists, and the parties best equipped to fix the underlying problems seem the least willing to do so.

Enterprises riding the Web 2.0 train should expect a rough ride ahead. While it is far too early to pull the emergency brake, IT managers should plan their journeys with care, recognize the terrain ahead, and seek opportunities to reduce the highest areas of risk. Specifically, enterprises should:

- **Scrub custom web application code** to prevent injection of malicious code and cross-site scripting (XSS). Restrict user input and sanitize all-out. To reduce business fraud caused by risk of cross-site request forgery (CRSF), enterprises should forcibly expire user sessions after 5 minutes of inactivity. An outside consulting firm should audit every public-facing application before going live.
- **Select third-party applications and toolkits with good security track records.** To reduce the risk of "forum poisoning" attacks on third-party and open source collaboration packages an enterprise uses, technology evaluators should carefully examine the security posture of products being considered. Desk research and scans of public mailing lists can help determine whether the development team of a particular package is proactive about security issues, or instead sweeps them under the rug. Services like Fortify's Java Open Review Project show how open source Java applications rank relative to each other in terms of security defect rates. Even though the project doesn't have full visibility into the web layer, the project provides some limited insights developers can use to make smarter security decisions.
- **Avoid PHP-based collaborative applications**, which are often riddled with programming errors and built on top of a notoriously insecure language (PHP). Packages written in type-safe languages (such as Java or ASP.NET) are safer, although they are not exempt from higher level logic and programming errors such as XSS.
- **Assess existing public-facing applications for security vulnerabilities** using an AVMS managed service like WhiteHat's, with standalone scanning tools from Cenzic,Watchfire (IBM) or SPI Dynamics (HP), or by retaining the services of an outside consultant.
- **Shield vulnerable public-facing applications** with a web application firewall from vendors such as Imperva and NetContinuum. These can help screen out XSS and SQL injection problems that enterprises don't have the resources, time or inclination to fix.
- **Block enterprise user traffic going to dangerous web sites** by using perimeter web-filtering products such as Finjan, WebSense or Secure Computing, or client-side products such as McAfee's SiteAdvisor. Web filtering products should be seen as palliative remedies only; they won't stop the most insidious attacks or fix root causes (defective server applications). However, these will partially protect enterprise users.
- **Delay deployment of high-value, customized Web 2.0 applications** until browser makers can demonstrate seriousness about fixing JavaScript and defining a true runtime security model.

VI. Further Reading

Yankee Group Link Research

Enterprises Heed the Roar of the Anywhere Consumer, Note, July 2007

Building the Developer Network, Part 2: Widgets and Mashups, Note, June 2007

Yankee Group's Anywhere Threats Research Protects Web-Based Businesses, Note, April 2007

Enterprises Heed the Roar of the Anywhere Consumer, Note, July 2007

Yankee Group

Yankee Group has research and sales staff located in North America, Europe, the Middle East, Africa, Latin America and Asia-Pacific. For more information, please contact one of the sales offices listed below.

Corporate Headquarters

Prudential Tower
800 Boylston Street
27th Floor
Boston, MA 02199
617-598-7200 phone
617-598-7400 fax
info@yankeegroup.com

Europe

55 Russell Square
LONDON WC1B 4HP
UNITED KINGDOM
44-20-7307-1050 phone
44-20-7323-3747 fax
euroinfo@yankeegroup.com

Yankee Group | the global connectivity experts™

A global connectivity revolution is under way, transforming the way that businesses and consumers interact beyond anything we have experienced to date. The stakes are high, and there are new needs to be met while power shifts among traditional and new market entrants. Advice about technology change is everywhere—in the clamor of the media, the boardroom approaches of management consultants and the technology research community. Among these sources, Yankee Group stands out as the original and most respected source of deep insight and counsel for the builders, operators and users of connectivity solutions.

For 35 years, we have conducted primary research on the fundamental questions that chart the pace and nature of technology changes on networks, consumers and enterprises. Coupling professional expertise in communications development and deployment with hundreds of interviews and tens of thousands of data points each year, we provide qualitative and quantitative information to our clients in an insightful, timely, flexible and economic offering.

Yankee Group Link

As technology connects more people, places and things, players must confront challenging questions to benefit from the changes: which technologies, what economic models, which partners and what offerings? Yankee Group Link™ is the research membership uniquely positioned to bring you the focus, the depth, the history and the flexibility you need to answer these questions.

Yankee Group Link membership connects you to our qualitative analysis of the technologies, services and industries we assess in our research agenda charting global connectivity change. It also connects you to unique quantitative data from the dozens of annual surveys we conduct with thousands of enterprises and consumers, along with market adoption data, comprehensive forecasts and global regulatory dashboards.

Yankee Group Link Research

As a Link member, you have access to more than 500 research reports and notes that Yankee Group publishes each year. Link Research examines current business issues with a unique combination of knowledge and services. We explore topics in an easy-to-read, solutions-oriented format. With the combination of market-driven research and built-in direct access to Yankee Group analysts, you benefit from the interpretation and application of our research to your individual business requirements.

Yankee Group Link Interaction

Our analysts are at your further disposal with data, information or advice on a particular topic at the core of a Link membership. We encourage you to have direct interaction with analysts through ongoing conversations, conference calls and briefings.

Yankee Group Link Data

Yankee Group Link Data modules provide a comprehensive, quantitative perspective of global connectivity markets, technologies and the competitive landscape. Together with Link Research, data modules connect you to the information you need to make the most informed strategic and tactical business decisions.

Yankee Group Consulting

Who better than Yankee Group to help you define key global connectivity strategies, scope major technology initiatives and determine your organization's readiness to undertake them, differentiate yourself competitively or guide initiatives around connectivity change? Our analysts apply Yankee Group research, methodologies, critical thinking and survey results to your specific needs to produce expert, timely, custom results.

Yankee Group Live!

The global connectivity revolution won't wait. Join our live debates to discuss the impact ubiquitous connectivity will have on your future. Yankee Group's signature events—conferences, webinars and speaking engagements—offer our clients new insight, knowledge and expertise to better understand and overcome the obstacles to succeed in this connectivity revolution.

www.yankeegroup.com

The people of Yankee Group are the global connectivity experts™—the leading source of insight and counsel for builders, operators and users of connectivity solutions. For more than 35 years, Yankee Group has conducted primary research that charts the pace of technology change and its effect on networks, consumers and enterprises. Headquartered in Boston, Yankee Group has a global presence including operations in North America, Europe, the Middle East, Africa, Latin America and Asia-Pacific.