Web 2.0 Article

Did you GET the memo?

Getting you from Web 1.0 to Web 2.0 Security

Author: Paul A. Henry

MCP+1, MCSE, CCSA, CCSE, CFSA, CFSO, CISSP, CISM, CISA, ISSAP, CIFI.
Vice President, Technology Evangelism, Secure Computing Corp.

The author would like to note appreciated contribution from Paul DeBernardi – Vice President Sales Enablement and Training

Table of Contents

Secure Computing® is a global leader in Enterprise Gateway Security solutions. Powered by our TrustedSource™ technology, our award-winning portfolio of solutions help our customers create trusted environments inside and outside their organizations.

Secure Computing Corporation

Corporate Headquarters
4810 Harwood Road
San Jose, CA 95124 USA
Tel +1.800.379.4944
Tel +1.408.979.6100
Fax +1.408.979.6501

European Headquarters
Berkshire, UK
Tel +44.0.870.460.4766

Asia/Pac Headquarters
Wan Chai, Hong Kong
Tel +852.2598.9280

Japan Headquarters
Tokyo, Japan
Tel +81.3.5339.6310

For a complete listing of all our global offices, see *www.securecomputing.com/goto/globaloffices*

The Evolutionary Web World

No, you did not miss the memo or a software upgrade notice. Yet, you've already arrived at Web 2.0.

The "upgrade" from Web 1.0 to the new Web 2.0 world has been an evolutionary process, continually driving the Web to be more interactive, useful and interesting for consumers and the business community. The evolution from Web 1.0 to Web 2.0 has been about improvements in the Web "experience"—from that of simply browsing static content and graphic images that display upon request, to an all-new highly interactive, programmable and much more useful Web.

Worse, the thing that you reach for to take off the virtual shelf may look like a book, but may not be a book at all; instead, it may be something else altogether—something disguised to look like a book, possibly a firecracker that explodes upon contact. The lesson? Beware. Be ready.

Oh, and incidentally, a much more dangerous place as well. Back in the early 1990s, our Web usage resembled that of a visit to a library in search of information or data. And, as far we know since libraries came into being, rarely has anyone ever been attacked by a book. In the new Web 2.0 world, however, the library is so real that the pages you're reading about lions seem to come alive—so much so that the next step may be that they actually jump out and take a bite out of you. Worse, the thing that you reach for to take off the virtual shelf may look like a book, but may not be a book at all; instead, it may be something else altogether—something disguised to look like a book, possibly a firecracker that explodes upon contact. The lesson? Beware. Be ready. To quote an old phrase, you cannot judge a Web page by its cover in the new World Wide Web.

The Web has long since evolved beyond simply serving up static content. It now incorporates a myriad of technology innovations such as highly interactive page content (things literally jump, spin, beep, and entertain us), real-time updates from RSS feeds, Weblogs, social networking sites, podcasts, and mash-ups. We've also embraced asynchronous programming languages and Internet protocols (i.e. AJAX) that dramatically enhance the users' interactive experience. In addition, new Web 2.0 technologies make it easy to actually contribute content, contribute attachments, and even contribute entire Web pages to Web sites like Wikipedia, YouTube, and MySpace, to name just a few Web 2.0 examples.

Web 2.0 Browsers Effectively Say "Throw it at me!"

When discussing Web 2.0 with everyday consumers, the easiest place to start is with Web browsers. Of course, Microsoft Internet Explorer rules that market—though there's healthy competition these days from Mozilla Firefox and Apple Inc.'s Safari.

In the Web 2.0 world, a browser is typically a willing player in the malware distribution game. Alas, the browser is a key tool in many of the underworld's schemes to compromise individual's privacy (e.g., Phishing). Quite often, the browser is used to infect machines with software that turns them into Zombie machines. Next up, the PC is illegally rented out as part of a cluster of Zombie machines called Botnets.

Botnets distribute huge volumes of spam email on command, or can be coordinated into a malware distribution networks.

Now, for the worst part: Botnets distribute huge volumes of spam email on command, or can be coordinated into a malware distribution networks, including being used for distributed denial of service (DDoS) attacks.

How is all of this possible? Unfortunately, there's a technology paradox at play here. Yes, a browser can be configured to reject unintended uses of the software. And yes, a browser can be set up so that it will not run executable programs, Flash video files, and so on. But if the browser is set up in that high security configuration, Web 2.0 and all of its great capabilities—animated graphics, RSS feeds, etc.—disappear.

But clearly, few people are willing to turn off Web 2.0 capabilities in their browser, even if they have no idea what Web 2.0 means. So what we have today is a world of browsers that will willingly display, store, or run anything you throw at them, and the underworld and hacking community knows that all too well. Oh my, what a dangerous Web 2.0 world we live in.

Few people are willing to turn off Web 2.0 capabilities in their browser

To sum up the situation, most browsers are like a two-year-old child crawling around the house putting anything in their mouth that they can get their little hands on. We all know how incredibly dangerous that would be for any infant who is not constantly supervised and protected. We would never let that happen in the physical world—but we do allow it to happen in the virtual browser world.

Risky Business

The risks don't end there. Today's hackers don't even have to compromise a Web site to inject malware for distribution; they are actually being invited to put anything on any Web site they want to. Talk about letting the fox into the hen house!

Today's hackers are actually being *invited* to put anything on any Web site they want to. Talk about letting the fox into the hen house!

So, what risks do we now face? Here's a prime example:
In November of 2006, Top Tech News (an Internet news and information site) reported that malware writers had used a Wikipedia article to lead users to a linked booby-trapped page. The page contained malicious code designed to plant viruses on the computers of unsuspecting users.

It was quite a creative ploy. The fraudulent Wikipedia page offered a bogus Windows security update for a version of the Lovesan/W32.Blaster worm, and included a link to an external site that was labeled with the name "*wikipedia-download.org*." The malware writers reportedly used the archive storage function on Wikipedia to plant the malicious code on multiple pages. The attackers simply directed users to those archived pages through emails that used the Wikipedia logo, and claimed that the encyclopedia site had been asked by Microsoft to help with worm patches. When the users clicked to get the helpful patch for the Microsoft vulnerability, they got the Lovesan/W32. Blaster worm instead.

The Journey to Web 2.0

Before we explore more benefits and risks associated with Web 2.0, it's important to understand how Internet applications have evolved. Wikipedia actually provides a very good timeline representation (Figure 1) of this evolution to Web 2.0.

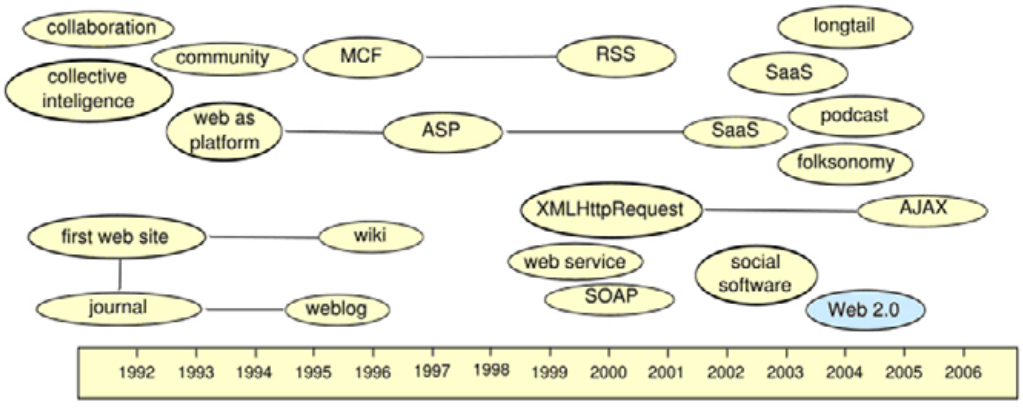

Figure 1: Web 2.0 evolution timeline. Source – Jürgen Schiller Garcia, Sept. 10, 2006.

Some would argue that the term Web 2.0 is nothing more than marketing hype being applied to technologies that have already been a part of the Internet for a long time, in some attempt to invigorate more interest in these technologies. Even so, there's a bigger picture to keep in mind. The significance of Web 2.0 is not any specific new technology like AJAX (Asynchronous JavaScript and XML). Rather, Web 2.0 represents a material change in the way both programmers and users of the Web (now and in the foreseeable future) will use it as a computing and networking platform. Most Web sites today incorporate at least some of the concepts and/or components of Web 2.0. And there are many widely-known examples such as Google, MySpace, Flickr, YouTube, Wikipedia, and Blogger that demonstrate the real capability and power that can be achieved by using Web 2.0 development techniques, software, applications, and tools.

The original concept of Web 2.0 has been credited to Tim O'Reilly and MediaLive International and was said to be the result of a brainstorming session that resulted in the first Web 2.0 conference. On May 16, 2006 the United States Patent and Trademark Office (USPTO) began allowing them to trademark the term Web 2.0 for use in their conferences.

In the initial brainstorming session of Tom O'Reilly and MediaLive International, they formulated a sense of Web 2.0 by example:

Web 1.0		Web 2.0
DoubleClick	→	Google AdSense
Ofoto	→	Flickr
Akamai	→	BitTorrent
mp3.com	→	Napster
Britannica Online	→	Wikipedia
personal Web sites	→	blogging
evite	→	upcoming.org and EVDB
domain name speculation	→	search engine optimization
page views	→	cost per click
screen scraping	→	Web services
publishing	→	participation
content management systems	→	wikis
directories (taxonomy)	→	tagging ("folksonomy")
stickiness	→	syndications

The original *Web 2.0 article* written in 2005 by Tim O'Reilly provides his highly regarded definition of Web 2.0.

What's Secure and What's Not?

When discussing Web 2.0 security, keep this dynamic in mind: many Web applications that carry security risks are not themselves insecure, but many sites using these programs become insecure when built with these capabilities.

To continue with our AJAX example mentioned previously, AJAX, the popular Web 2.0 programming language, itself is not insecure but many insecure Web sites are today built with AJAX.

Over time, we have vetted the majority of security issues with the underlying protocols for Web 1.0. Today, however, the layering of new next-generation programming languages on top of these protocols in Web 2.0 has given the Internet's bad guys a whole new set of opportunities to exploit. A great example of this would of course be AJAX. The asynchronous nature of AJAX, which allows dynamic page activity driven by user choices, clearly improves the users experience on a Web site by taking interactivity to an entirely new level. However, because of the unpredictability aspect of this, it also dramatically increases the chances that things can go terribly wrong from a security perspective.

Here's why: Older synchronous programming languages restricted interaction to that of a defined and orderly format—safeguarding us all from security chaos. **In stark contrast, AJAX operates asynchronously, whereby actions do not necessarily follow a defined orderly format. The result: it's nearly impossible to fully vet out any potential bugs that could result in security issues.** All of the risks associated with any programming language (such as race conditions, code correctness, object violations and incorrect error handling) are amplified significantly when operating in an asynchronous environment such as that provided by AJAX.

The mere use of AJAX in Web 2.0 applications can increase the possible threat envelope due to the increased interactivity with the user's browser. Further, if the Web site-based AJAX program also needs to interact with JavaScript that runs on the user's browser, an additional security risk is now added to the risk equation.

Some pundits argue that AJAX by-and-of-itself is not at fault and that AJAX does not increase the threat envelope. Instead, they would argue, the real issue is AJAX programmers. By using AJAX to increase application functionality, the programmer theoretically increases the possible number of server-side vulnerabilities. Our purpose here is not to focus on fault, or to say risks stem from so-called "bugs" within any given program, including AJAX. Rather, the point is that the nature of the interactive capabilities these programs provide themselves open the door to risk and increase the possible threat envelope….so, users beware.

One final point on AJAX: It is relatively new and secure programming standards have not yet been fully vetted. **As a result, traditional Web site vulnerabilities to attacks like XSS (Cross Site Scripting) could start re-appearing in Web 2.0 Web sites.**

Consider this: The worm that was used to attack MySpace (enabled by AJAX), as well as the worm that was used to attack Yahoo! (also enabled by AJAX) both took advantage of a component of AJAX called XHR to help propagate the worms.

Déjà Vu – Are We Building Yet Another Bubble?

Web 2.0 certainly allows us all to innovate on the Internet. Unfortunately, just as what happened in the early 1990's Internet boom, businesses and individuals are rushing the deployment of these new Web capabilities and features with little, if any, regard to security.

Hence, we find ourselves in a position now, due in large part to rushed Web 2.0 implementations, that the Internet is a much more dangerous place to be than it ever has been. Web-based email providers, photo sharing Web sites, blogs, Wikis, and social networking sites have all fallen victim to malicious hackers due to their lack of consideration of security in the "new" Web 2.0 world.

Internet Threat Vectors

In their quest to harness the power of the Internet, enterprises began increasing the connectivity of their internal applications to the Web. The threat vector originally involved layer 4 (the network layer) of the OSI model, where inspection is primarily limited to IP address and port numbers in Stateful packet filters. But the threat vector soon shifted to layer 7 (the application layer), where attackers could exploit vulnerabilities of Internet-connected applications.

Now, for the problem: As the threat vector shifted from layer 4 to layer 7, our defenses simply did not keep pace.

With the change in the threat vector, signatures for known attacks began to find their way into firewall security products. Some stateful packet filter vendors attempted to offer at least some level of application-layer attack protection. This protection methodology is often called a Negative Security Model, whereby all traffic is allowed to flow freely and the protective mechanism uses the signature of known attacks layered on top of their stateful packet filters. This approach attempts to enumerate potentially malicious traffic and to block it only once (and if) it has in fact been identified.

Unfortunately, the Negative Security Model is only reactive in nature.

Unfortunately, the Negative Security Model is only reactive in nature. Admittedly, these products are marketed as being proactive because of their ability to automatically block an attack on behalf of the product user. However, a signature for a given attack must first be created before any defense against that particular attack can be afforded. As a result, the use of this methodology in reality is not at all proactive and is at best only a reactive methodology. In today's environment, where over 6,000 application vulnerabilities are reported annually, vendors are having a difficult time maintaining defensive signatures for these known attacks.

But what about all the unknown threats circulating across the Web? A recent study found that the typical vulnerability exists for up to 348 days before public disclosure. Hence the malicious hacker who found the vulnerability could potentially have nearly a year of free reign to exploit a vulnerability before a defensive signature can be created.

But what about all the unknown threats circulating across the Web? A recent study found that the typical vulnerability exists for up to 348 days before public disclosure.

The problems don't end there. In a *recent article*, IBM warned that there is a colossal difference between the number of vulnerabilities disclosed publicly and the number of vulnerabilities that are discovered and are not publicly reported. IBM has estimated that up to 139,362 vulnerabilities are discovered annually—but not reported publicly.

IBM has estimated that up to 139,362 vulnerabilities are discovered annually—but not reported publicly.

Remaining Application-Layer Risks with Web 2.0

Clearly, the increased functionality of Web 2.0 Web sites along with the relatively new underlying programming languages are creating new threat vectors and revitalizing traditional threat vectors. The most common and "most concerning" threat vectors for Web 2.0 include:

- Web-borne malware
- Real-time RSS/Atom Feeds with JavaScript Malware inside
- XSS Scripting (Cross Site Scripting) - e.g., MySpace Worm
- CSRF (Cross Site Request Forgeries) - Stealing data in Java space, e.g., Gmail
- XSS filter bypassing - ENCODING
- Exponential XSS Attacks - No need to limit to 1 Web site
- Forging "request headers" using Flash
- Backdooring Media Files - JavaScript in everything

Web-Borne Malware

Web-borne malware exploits a blind spot in many security implementations today. Specifically, many applications fail to inspect the data returned from a visit to an Internet Web site. Typically using Java Script, malicious hackers literally append malicious code to the Internet Web page. **When the unsuspecting user visits the infected Web page, the Java Script runs in the user's browser and in most cases causes malware (such as a key logger or root kit) to be downloaded to the user's PC.** This methodology has also been referred to as "**Drive By Hacking**" and has impacted several high-profile Web sites. One prime example: At the peak of the NFL playoffs, hackers compromised the Miami Dolphins' Web site and malware was automatically loaded onto visitors' PCs. Similarly, the Center for Disease Control Web site was compromised, and unsuspecting visitors had malware loaded onto their PCs.

The most notable Web-borne malware event to date involved the Storm worm and its respective spam botnet. Starting in January 2007, the Storm worm has gone on to impact hundreds of

thousands of users. (Read our *Storm Worm Threat Alert*) The malware generated over 9 million emails to users. Each message directed recipients to Web sites containing malicious code. While the social engineering aspects change regularly, the Web 2.0/Web-borne Malware delivery methodology remains the vehicle of choice for the delivery of its root kit/spam bot malware.

Still in Denial

Even today, Web server administrators irresponsibly dismiss Web-borne malware as non-malicious attacks, while thousand of their Web site visitors are put at risk. An estimated 450,000 URLs point to malicious Web sites hosting malware on the Internet. That's nearly half-a-million hidden landmines that casual users and even advanced users can't spot on their own.

RSS/ATOM Feeds

Nearly 12% of Internet-connected users take advantage of RSS/Atom feeds to receive timely news and data content. Alas, most users don't consider the security ramifications of connecting to a remote server for an RSS/Atom feed. To be sure, using RSS to deliver malware is well within the realm of possibilities.

The RSS threat isn't new. As far back as 1995,Yahoo was alerted to an RSS feed vulnerability in the company's RSS aggregator.

A more recent security issue has been found with the AOL ICQ Toolbar (CVE-2006-4660). In this example, the default *options2.html* Web page is not validated before loading. This permits a hacker to execute arbitrary script/coding by tricking them into visiting a malicious Web site. A second vulnerability in AOL ICQ Toolbar (CVE-2006-4661) takes advantage of a failure to validate the title and description fields of the feed. This sets the stage for a hacker to trick a user into visiting a malicious Web site to execute arbitrary HTML/scripting.

XSS Scripting

XSS allows malicious hackers to inject code into the Web pages viewed by others. This methodology has become popular in Web 2.0 Phishing exploits and browser vulnerabilities. In some cases, it allows the hacker to bypass access controls within the user's network.

There have been several XSS exploits on the Internet and two received a great deal of media attention:

- MySpace XSS Worm
  - *A worm written to exploit* an XSS vulnerability in MySpace brought the service down for nearly two days. The exploit injected JavaScript into users' pages and when the Web page was visited the JavaScript was executed in the visitors' browsers.
- Yahoo XSS
  - *XSS vulnerability in Yahoo* tricked users to click on a booby-trapped link. After each user clicked on the link, the hacker gained access to the users' Yahoo account including email, address book and calendar entries.

It should also be noted that a security researcher recently uncovered 40 flaws in Google's YouTube Web site. The vast majority of these flaws were XSS issues that put users at risk of having their profiles infected with a fast-spreading worm that could potentially steal users' credentials.

Cross-Site Request Forgeries (CSRF)

CSRF exploits are often confused with XSS exploits. However, CSRF does not rely on any client-side active scripting. Instead, CSRF exploits a victim's prior relationship and authority with a Web site to allow unwanted or unapproved actions by a malicious hacker.

Several CSRF exploits have been reported. One particular CSRF vulnerability in phpMyAdmin (CVE-2006-5116) could allow a remote attacker to perform unauthorized activities posing as another user simply by setting a token in a specially crafted URL. Another CSRF issue demonstrated at the

2006 BlackHat conference showed that a user's DSL router configuration could easily be changed because the user's browser automatically supplied the DSL router with the user's credentials—even though the hacker (NOT the user) was accessing the DSL router.

CSRF exploits in security products are not limited to DSL routers. In an article at Calyptix, CSRF exploits were found to exist in at least four mainstream security appliances, allowing a remote user to gain administrative access.

XSS Filter Bypassing

With the large number of XSS issues plaguing the Web 2.0 world, administrators began utilizing filters to ward off XSS attacks. Filtering to block XSS attacks can be effective but is often found to be weak when the Web site must support multi-national users. Simply put, most XSS filters are written to support a given character set for a specific language. To bypass the XSS filters, hackers simply switched the default language encoding (i.e. UTF-8 to US-ASCII).

XSS filter bypassing issues have been found in several software products including PHPNuke and the firewall product from NetGear (FVS318) that allowed hackers to run XSS attacks against FVS318 administrators.

Exponential XSS Attacks

While a traditional XSS attack targets an individual user, an exponential XSS attack targets hundreds or perhaps thousands of users. Proof of Concept code for an exponential XSS attack can now be found on the Internet.

The example attack, known as the"Nduja Connection," is a worm that targets Web-based email systems. It collects the email of a given user, collects all of the user's contact email addresses and then self propagates to all of the users' contacts. This exponential XSS attack was used successfully against *Liberi.it, Tiscali.it, Licos.it*, and *Excite.com* Web-based email systems.

Forging Request Headers Using

Flash, the popular browser add-on from Adobe, has been found to facilitate the sending of HTTP request headers from within the scripting used in its Flash Action Script. The ability of a hacker to cause a victim's browser to send HTTP requests to third-party Web sites has many serious security implications:

Browsers in general are limited by how large a potentially malicious payload can be sent in a forged request header. This browser limitation was thought to have reduced the risk of the recent Apache Expect Header vulnerability. Malicious hackers quickly figured a way around the browser limitation by packaging the forged requests into a Flash file. The specially coded Flash file delivers the full malicious payload without the size restriction of a typical browser. Hence, the risks associated with the Apache Web server Expect Header vulnerability were greatly increased as a reliable delivery vehicle was found in Flash.

The ability to forge HTTP request headers has been successfully demonstrated with Microsoft IE 6.0 as well as Firefox 1.5.0.4 while running either Flash 7.0X or 8.0X.

Backdooring Media Files

Using the scripting capability of one product to exploit a vulnerability in another is nothing new in our Web 2.0 world. We have seen this used successfully with Flash and QuickTime and have seen a theoretical backdooring exploit with PDF files. However, recently we have seen a move to backdooring even more popular media file types—MP3 files—with proof of concept code.

Risk Mitigation Considerations in a Web 2.0-3.0 World Protecting Internet-Facing Applications

There are some natural first steps for protecting your applications. For starters, you need to establish security policies that define specifically what can be exposed to the public Internet.

Secure coding of Internet applications is the first layer of defense. Adopting secure coding practices in development is mandatory; the granular and exhaustive testing of Internet applications is more important than ever. In the early days of programming, you could manually step through combinations of user actions to validate your security settings.

But we have transitioned from historical programming languages and environments that were sequential in nature. These days, AJAX and other asynchronous approaches that make manual testing almost an impossible task. In a synchronous environment it was simple to test defined paths that users followed; actions were predictable and could be easily tested. In an asynchronous environment, there are perhaps too many possible combinations of possible user actions for traditional manual testing methodology.

In our Web 1.0 world, an administrator controlled the Web sites' content. In our Web 2.0 world, the Web site content is a combination of administrator and the many people that visit the site. Hence, it is critical that any and all of that user-contributed content is fully tested before being either run on the Web server or made available to the public. For instance, Web systems must use URL filtering and a reputation-based anti-malware component to protect the Web site as well as Web site visitors.

The use of an application-layer firewall (not to be confused with an application-layer filter) is a mandatory requirement in the protection of Internet-facing applications. Application-layer firewalls go a long way in reducing the threat envelope. Specifically, they reduce the effective command set available to a remote user. Think of it as reducing the number of tools a hacker can use to penetrate your systems.

Current application-layer firewalls also include the use of signatures as a second layer of defense. These specially crafted signatures for entire classes of attacks are compared with traffic to further enhance risk mitigation. By using signatures that depict classes of attacks, application firewall vendors use signatures of specific attacks in their effort to filter their way to security. **Simply put, a minor change in the attack that alters its signature, does in fact make the signature provided by the application-layer filter vendor completely useless in preventing the attack.**

Lastly, visitors can post a malicious script on the Web server. It's therefore critical to isolate the Internet-facing server with the policies of the application firewall to limit the ability of the server hosting the Internet application from freely opening connections to servers within the network that resides behind the Web server. The evolution to Web 2.0 has increased the risk that a compromised Internet-facing server could be used as a launching point for deeper attacks within the enterprise network.

Protecting Users within the Enterprise Network

Internet right-of-use policies have to be updated to include the organization's position on the use of new Web 2.0 technologies—such as but not limited to Internet blogs, Wikis, and social Web sites.

Beyond the policy updates, technical safeguards are mandatory. Gone are the days when an organization's internal user Web security amounted to nothing more than Internet the configuration of a firewall that simply opened port 80 for all internal users to surf the Web. All traffic across all applications into the internal network must be inspected for malicious content, especially that from employee-generated traffic.

Anti-Malware as a Component of Web 2.0 Risk Mitigation

Anti-malware scanning eliminates the blind spot found in traditional anti-virus products that only rely on signatures of know attacks for risk mitigation. Scanning or scrubbing traffic first requires the normalization of that traffic to defeat the encoding and obfuscation tactics of malicious hackers. Once normalized, the traffic must be inspected for scripts (such as Java and active X) that may be encapsulated to determine if in fact those scripts are of malicious intent. Simply put, any traffic containing malicious scripts must be blocked at the gateway and the blocking of future access to the Web site that returned the malicious scripts should be considered.

Anti-malware scanning has already "field proven" its effectiveness in our Web 2.0 world. The malicious code in the Web site-borne malware used in highly publicized attacks involving the Miami Dolphins, CDC, Korean gaming, and other Web sites were fully mitigated. The malicious code associated with the Storm worm including the recent greeting card spam was fully mitigated. The embedded malware in the QuickTime files used in the MySpace XSS as well as that used with the ICQ Tool bar exploit and Yahoo XSS was fully mitigated. The current growing trend in embedding malicious scripts within media files including the recent POC code recently released that incorporated malicious Java Script within PDF files was fully mitigated.

Further, it is not uncommon for current technology anti-malware products to incorporate complementary technologies such as Header Filters and Cookie Filters. These, respectively, mitigate the Web 2.0 risks associated with attacks based on the Apache Expect header vulnerability whether initiated from a browser or initiated from code embedded within a media file and to mitigate the risk of CSRF related data theft attacks.

Reputation-Based Defenses as a Component of Web 2.0 Risk Mitigation

Reputation-based defenses are quickly becoming a mandatory component of enterprise Web 2.0 risk mitigation. Today malicious hackers very often utilize a compromised PC, especially those within Botnets for multiple nefarious purposes. Sending spam, hosting malicious Web sites that automatically pass malware along to site visitors, and supporting the distribution of malicious content such as compromised video, document, and other socially acceptable files are now normal activities of these compromised PCs. Because of these activities, it is possible to quickly develop a reputation score for the individual PC as it interacts with others across the Internet. These reputation scores can afford tremendous risk mitigation to those organizations taking advantage of reputation-based defenses.

URL Filtering with a Reputation Element as a Component of Web 2.0 Risk Mitigation

Reputation-based defenses, when used to complement traditional URL filtering, affords a great deal of additional risk mitigation in our Web 2.0 world. Specifically, they block internal user access to an Internet-based server that has a reputation beyond the risk acceptance of the enterprise. Google recently tested 4.5 million URLs and found that 450,00 of those tested (1 in 10) had some form of a malware component. It is not uncommon for nearly 10,000 malicious Web sites to be identified based upon the reputation of their malicious activities on a given day.

Anti-Spam with a Reputation Element Component of Web 2.0 Risk Mitigation

It is important to recognize that malicious hackers are now including links within socially engineered emails to trick users in to clicking on URL links. These links direct the email recipient to Internet-based Web site servers that host Web site-borne malware. Hence, even traditional spam or targeted email in the enterprise email system has quickly emerged as a component of Web 2.0 threats.

Now, for the key takeaway: Reputation-based defenses (when used to complement traditional anti-spam filtering) affords a great deal of risk mitigation by comparing the senders' IP address or network with the reputation database to determine the probability that the email is spam. This approach can reduce spam by up to 90%. The Internet has been flooded for many months now with spam in the form of fake greeting card announcements (Figure 2) that contained a URL directing the receiver to a malware Web site, hence taking action based on the reputation of an email (Figure 3) a reduction in spam can also afford Web 2.0 risk mitigation as well.

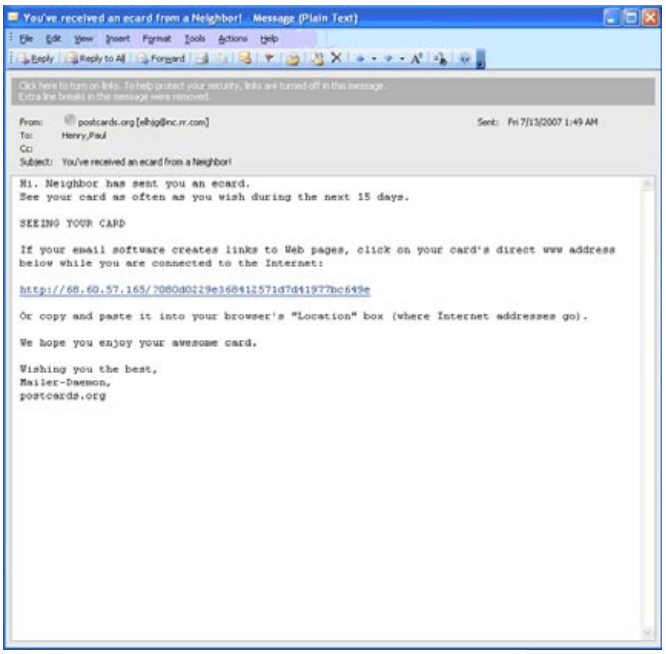

Figure 2: Fake greeting card announcement

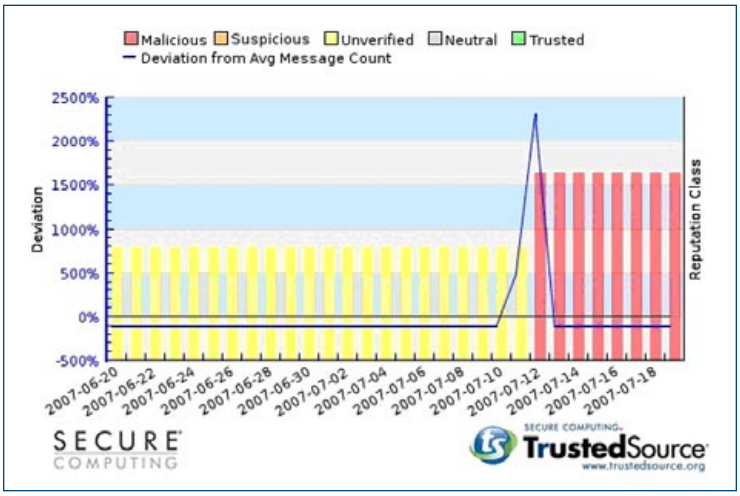

Figure 3: TrustedSource reputation data shows malicious activity

Firewalls with a Reputation Element as a Component of Web 2.0 Risk Mitigation

Reputation-based defenses (when used to complement traditional firewalls) affords a great deal of risk mitigation in blocking access to network assets from high-risk sources. Further, one of the favorite tactics of malicious hackers today is to use the compromised PCs from their Botnets in their Distributed Denial of Service (DDoS) attacks. The goal of these attacks is to tie up network resources with half open connections and even with what appear to be legitimately completed connections.

Authentication with a Reputation Element as a Component of Web 2.0 Risk Mitigation

Reputation-based defenses (when used to complement traditional authentication mechanisms) affords a great deal of risk mitigation in blocking authentication requests from connections originating from IP addresses or networks that have a reputation beyond the risk acceptance of the enterprise. Malicious hackers today are using the compromised PCs in their Botnets in both theft of user credentials as well as even more advanced credential theft, using Man-In-The-Middle (MITM) attacks. It is prudent in our current Web 2.0 environment to address this threat—perhaps by considering blocking authentication requests based upon the reputation of the requesting IP address or network sending the authentication request.

ID Theft Is Being Fueled by Web 2.0 Insecurities

The fastest growing crime in America—ID Theft is being fueled by Web 2.0 insecurities. *Targeted attacks* against key executives using socially engineered email messages, driving them to open malicious email attachments that automatically install key loggers have become commonplace.

A recent report from Gartner clearly shows that malicious hackers have, in fact, become much more targeted. Gartner found that if you earn more then $138,000 per year, you will receive 50% more spam, and higher income individuals tend to lose more money when they fall for a scam as well. For example, if you earn less then $138,000 and fall victim to phishing, the average loss is $1,500, while those earning more than $138,000 lose $5,700 on average.

So how do these attacks begin and evolve? Let's take a look:

First, a BBB Phishing Trojan...

The most recent example of this was a phishing scam that started as a targeted attack against executive-level managers. The mass email included a subject line indicating the message contained a consumer complaint from the Better Business Bureau (BBB). If a recipient clicked the attachment, a sophisticated Trojan was installed on the recipient PC that stole all interactive data sent from the recipients' Web site browser to a compromised Web site server.

Morphed into IRS Phishing...

As with the BBB email, a new version of the phishing Trojan was disguised as a criminal investigation notice from the IRS. This was also a targeted attack against executive-level managers and contained a similar—if not identical—malware payload. When the user clicked the attachment, a sophisticated Trojan was installed on the recipient PC that stole all interactive data sent from the recipients' Web browser. With the IRS email, the malware launcher set up a new server to receive the stolen information that was registered to a domain in China. The server was also physically located in China.

Reverted Back to BBB Trojan

The second time around, the email scam used a domain called "business-compalints.com," registered in China. This second version was thought perhaps to be a more convincing message due to the domain name.

Change in Tactics: FTC Camouflage

In this scam, hackers used a spoofed FTC email address designed to convince the recipient that someone had filed a complaint against them. A copy of the complaint was attached to the email—but the attachment contained the Trojan. Again, the email targeted executive-level managers. When the recipient clicked to view the attachment, a sophisticated Trojan was installed on the recipient PC.

Next, the Trojan stole all interactive data sent from the recipients' Web browser.

New "Proforma Invoice" Disguise

The current version of the scheme is still operating as a targeted attack aimed at executive-level managers and is again using a Trojan. The Trojan steals all interactive data and sends it from the recipients' Web site browser to one of three domains: (1) *www.hlplace.com*, (2) *www.tanzatl.org,* and (3) *www.aecv.ch*. This scheme uses so-called social engineering—which is the hacking of a normal human process or occurrence. Specifically, this scheme uses a Proforma Invoice, the receipt of which, by a senior level manager, is not an uncommon occurrence.

So, who's behind these schemes? There is some disagreement within the research community. Some researchers believe there is a single group of coordinated hackers at work. Others think competing groups are each learning from the others' success and adding the new wrinkles to the next variant of the scam. Traffic analysis and reverse engineering the Trojans revealed the data was being sent to servers in China or servers related in some way to China but anyone could have just as well compromised China-based servers for use in exploits to throw researchers off. The Trojans sent stolen information to China either through servers registered and located within China or through compromised servers in other countries thought to be controlled by hackers in China.

As of June 2007, only a small number of anti-virus vendors were able to detect the payload of the Proforma Invoice email as being malicious. For unsuspecting users, these emails are quite well written. They typically include a plausible subject line and a well-socially-engineered message.

Since these are very targeted messages rather than bulk spam blasts, anti-virus will not be able to detect the payload and this scam will likely be one of the most profitable in recent memory. The most effective defense is of course a multi-layer defense:

- The use of an anti-spam solution with a reputation element to drop email based upon the senders reputation
- URL filtering with a reputation element to block connections to the malware hosting Web sites
- Anti-malware scanning to block the malicious Java Scripts either embedded in email attachments or returned to the users Web site requests that download and execute the malicious payload
- Data leakage protection that blocks any personal information from being inadvertently sent out of the network over any protocol

In Closing

Enterprises have forged ahead with the rapid evolution from Web 1.0 to Web 2.0 without addressing the inherent security risks. It is imperative for organizations in the Internet age, in order to remain competitive, continue to embrace new technologies such as many of those offered by Web 2.0. However, security must shift from being that of an after thought to that of a primary design consideration. Plus, it must be implemented long before enterprises expose their Internet-facing applications to the inherent risks of Web 2.0. Further, it must be considered that even those organizations that have not yet planned or implemented their own Web 2.0 application services are still at risk to the many Web 2.0 threats that their internal users are currently exposed to in simply surfing today's Internet.