

## Web Browser Security *Deep Dive*



## How to stay secure on the Internet



## An expert's guide to Web browser security

Today's Web browsers have different security pros and cons, and none offer a magic bullet against threats. Here's how to keep your Web surfing secure.

By Roger A. Grimes

WHICH WEB BROWSER is guaranteed to make your Internet browsing experience perfectly safe? The answer is none, of course. If you have the need for high security on a computer you manage, then you shouldn't allow it to surf on the public Web. It's that simple. But if your need for security is not extreme, there are a number of things you can do to make your Web browser more secure and your Web surfing safer. Let this Deep Dive be your guide.

Internet browsers are highly complex pieces of software that interact with highly complex programming code, much of it not so friendly. There is no "super secure" browser. The number of known exploits against a particular browser exactly tracks to its popularity. No surprise there. Even secure alternatives to Internet Explorer, which all new browsers seem to claim to be, generally have been targeted by dozens of exploits. (Even the newest of these, Google Chrome, already has a dozen.)

Today, a significant portion of computer attacks comes from legitimate Web sites that have been maliciously modified. In short, limiting your surfing to only well-known, legitimate Web sites does not ensure a safe Internet browsing experience. And the problem will only get worse, not better, for the near-term future.

### BROWSER SECURITY WARS

About a year ago, I spent several months running the five most popular browsers – Internet Explorer, Firefox, Google Chrome, Safari, and Opera – through a battery of security tests. Much to my surprise, *none* of the browsers allowed malware to silently install on my test systems. In other words, if a fully patched browser is running on a fully patched Windows system (Windows

XP Professional SP3, in my tests), then malware's best chance of success is fooling the user into willingly executing it. This is why socially engineered Trojan horses – fake browser plug-ins, fake anti-virus programs, etc. – are so common. Beware.

Yes, there will always be zero day exploits that can silently infect through a browser, but in testing, I found out that on every malware site that I visited (and I am confident that it was a good representative sample) each offered up an executable to install or tried to use an exploit for software that had already been patched. Using a fully patched system (all software, not just the browser) prevented all silent attacks in my real-world tests.

I spent weeks looking for zero day exploits to test against, and by the time I found the sites, they had been taken down or the hole had been patched. This is not to say that zero day exploits won't get some people. Obviously, they do. But they are a very small minority. The average end-user is far more likely (say, 99.999 percent) to come across exploits trying to leverage holes that have patches available.

Almost all the malicious Web sites I came across offered an executable to install, usually in the form of bogus anti-malware software or some sort of content player. In order to be infected, I had to intentionally run the offered executable – not always, but nearly so. There was a smattering of sites that tried to use malformed or mismatched content to trick the third-party software into silently executing code, but it was uncommon; and when my system was fully patched, it never silently succeeded. The converse is also true. When I intentionally installed the offered malware, every browser allowed the underlying host system to become compromised.

The results back up everything I've been saying for the past few years. Your best defense against malicious



attacks is a fully patched system (OS, browser, browser add-ins, and all other software), and educating your end-users to not install the bogus offered executable (which can often look very legitimate).

Nearly all real-life exploits use JavaScript to launch the executable. It's easy to disable JavaScript support in all the browsers, except for Chrome, but doing so can also cause problems with a high percentage of legitimate Web sites (throwing the baby out with the bathwater). Disabling JavaScript makes sense when an unpatched zero day is launched and spreads rapidly (it does happen occasionally). But most serious zero day exploits are patched within a few days, so the days of risk exposure are minimized.

Another interesting result of my browser security reviews: I was surprised by how many security features each of the browsers shared (anti-phishing, cookie control, anti-XSS handling, pop-up blocking, file download detection, digital certificate handling, and so on). Each browser also presents certain strengths that will appeal to different users. I'll explore these security features in detail below.

## MAKING A SECURE BROWSER

Many security pundits recommend any browser but Internet Explorer as the best security defense. Although there is some safety in using less frequently attacked software, a better question is which is the safest choice among the most popular browsers? What are the most important security features to look for in a browser, and what are the weaknesses to beware?

Each new browser entry typically promises a more secure browsing experience, only to prove that making a truly secure Web browser is difficult. Each of the most popular browsers has dozens of patched vulnerabilities. Even the newest, Google's Chrome, released in beta form in September 2008, has [more than a dozen exploits](#). Perhaps the strongest testament to how hard it is to make a secure Internet browser is the fact that even the text-only [Lynx browser](#), which is as simple as a browser can be (it can't even display pictures or video without external programs), has had [five vulnerabilities](#). If attackers can cause buffer overflows in a text-based browser, any browser more complex will have its issues.

In general, administrators must consider every Internet-connected Web browser as high risk. In very high-security environments, Web browsers aren't allowed to

run or aren't allowed to render content from the Internet. But assuming your enterprise needs to browse the Internet and seeks a Web browser with an acceptable level of security, keep reading. A secure browser must include the following traits as a minimum:

- It was coded using Security Development Lifecycle (SDL) techniques.
- It has undergone code review and fuzzing.
- It logically separates network and local security domains.
- It prevents easy malicious remote control.
- It prevents malicious redirection.
- It has secure defaults.
- It allows the user to confirm any file download or execution.
- It prevents URL obscurity.
- It contains anti-buffer overflow features.
- It supports common secure protocols (SSL, TLS, etc.) and ciphers (3DES, AES, RSA, etc.).
- It supports Extended Validation, or EV, digital certificates. Browsers that support EV certificates display a special icon, or shade the address bar, when a user surfs to a site secured by one.
- It patches and updates itself automatically (with the user's consent).
- It has a pop-up blocker.
- It utilizes an anti-phishing filter.
- It prevents Web site cookie misuse.
- It prevents easy URL spoofing.
- It provides security zones/domains to segregate trust and functionality.
- It protects the user's Web site logon credentials during storage and use.
- It allows browser add-ons to be easily enabled and disabled.
- It prevents mischievous window use.
- It provides privacy controls.
- It has been battle tested by hackers over a sufficient period of time.

Another good place to start learning the detailed basics of Web browser security is [Part 2 of the Browser Security Handbook](#), maintained by Michal Zalewski. The Browser Security Handbook gives a great introduction



to many of the behind-the-scenes security policies that underlie most of today's browsers and indicates which features are supported in various browsers.

## HOW TO MEASURE THE SECURITY OF A BROWSER

Vulnerability counts and the frequency of announced exploits account for much of the overall risk to a Web browser, but they are far from the only relevant factors to consider. In the security review, the following criteria were used during evaluation:

**Security model.** Each browser is coded on the underlying strength of the browser vendor's chosen security model. This model is what keeps the untrusted network side separated from the more trusted security zones. If malware is able to exploit the browser, how easily can it compromise the whole system? What defenses did the vendor include in the browser's underlying design to prevent malicious use? How is malicious redirection (such as cross-domain cross-site scripting and frame theft) prevented? Is memory secured and cleared against malicious reuse? Does the browser give end-users multiple security domains or zones with varying levels of functionality in which to place different Web sites according to their level of associated trust? What end-user protections have been built into the browser? Does the browser attempt to update itself? All of these questions, and more, go into determining the fitness of a browser's security model.

When the browser runs on Windows does it take advantage of Data Execution Prevention (DEP)? If it runs on Windows Vista, does it use file and registry virtualization, Mandatory Integrity Controls (see below), or Address Space Layout Randomization? These topics require too much space to discuss appropriately in this review, but all four mechanisms can make it harder for malware to gain system control.

**Feature set and complexity.** More features and increased complexity are the antithesis of computer security. Additional features mean more code available to exploit with more unexpected interactions. Conversely, a browser with a minimal feature set may not be able to render popular Web sites, which forces the user to use another browser or to install potentially insecure add-ons. Popular browser add-ons are often exploited by malware writers.

User-definable security zones (also known as security domains) are also an important feature. Ultimately, less functionality translates into better security. Security zones provide a way to classify various Web sites as more trustworthy and, hence, suited for greater functionality. You should be able to trust your company's Web sites significantly more than a Web site offering pirated software or a small Web page served up by someone you don't know. Security zones allow you to set various security settings and functionalities based upon the Web site's location, domain, or IP address.

Security domains are used in every computer security product (firewalls, IPSes, and so on) to establish security boundaries and areas of default trust. Having a security zone in a browser extends that model. Browsers without security zones encourage you to treat all Web sites with the same level of trust -- as well as to reconfigure the browser or use another browser for less trustworthy Web sites before each visit.

**Vulnerability announcements and attacks.** How many vulnerabilities have been found and publicly announced against the browser product? Are the vulnerability counts going up or down as the vendor patches its browser? How severe have the vulnerabilities been? Do they allow full system compromise or denial of service? How many vulnerabilities are currently unpatched? What is the history of zero-day attacks against the vendor? How often is the vendor's browser targeted versus a competitor's product?

**Browser security tests.** How did the browser fare against popularly available browser security test suites? In this review, all of the products passed the most well-known browser security tests located on the Internet, so each item was further exposed to dozens of real-life malicious Web sites. Often the outcome was not pretty. I experienced frequent browser lockups, objectionable content, and sometimes complete system reboots.

**Enterprise manageability features.** InfoWorld caters to administrators and technicians who need to accomplish tasks across an entire enterprise. It is generally easy to secure a favorite individual browser for personal use, but doing so for an entire business requires special tools. If the browser were selected for enterprise use, how easy is it to install, set, and manage secure configurations for every user?





These are the general categories that I considered when reviewing each Internet browser.

## HOW I TESTED

I downloaded the latest publicly available version of each browser (including beta products) and installed it on fully patched 32-bit versions of Windows Vista Enterprise SP1 and Windows XP Pro SP3. I reviewed all security settings and options and checked the vendor documentation for clarification. I then subjected each browser to numerous tests, including dozens of pre-defined tests made in the lab, Internet-based test suites, and exposing the browsers to known-malicious Web sites.

The Internet-based test suites included several browser security test sites, such as [scanit](#) and [Jason's Toolbox](#); several JavaScript, Java, and pop-up blocker testing sites; several cross-site scripting (XSS) testing Web sites; and several browser privacy test sites. I tested the security of the browsers' password handling using the [Password Manager Evaluator Web site](#) and the security of cookie handling using the Gibson Research Corporation's [Cookie Forensics Web site](#). I tested Extended Validation certificates using links provided on [the IIS7 site](#).

I surfed to dozens of Web sites known to contain live malware from several public and private malware site lists, including [ShadowServer](#). I also visited dozens of known phishing Web sites, courtesy of [PhishTank](#) and similar referral sites. I used [Process Explorer](#) to monitor local processes and resources during install and ongoing operations. And I sniffed the browsers' network traffic using [Microsoft Network Monitor](#) or [Wireshark](#) and examined the results for information leaks.

Finally, I also relied on public vulnerability testing for these evaluations, including [Metasploit](#) and [milw0rm.com](#). Vulnerability statistics were taken from [Secunia.com](#) or [CVE](#).

Additionally, each browser was used over a series of several weeks (or longer) to test general use, patching intervals, and other involved functionality.

## THE MOST SECURE BROWSER

Which of the browsers tested can claim to be the most secure? Here's the big shocker: None of the fully patched browsers allowed silent infections or exploitation beyond simple DoS attacks. All of the browsers stopped the latest malicious attacks available on the

Internet. Occasional zero-day attacks could silently infect a particular browser during a particular period of time, but all of the browsers have this same risk, and all of the browser vendors in this review are fairly consistent in patching significant problems in a timely manner.

Hence, the overall conclusion of this review is that any fully patched browser can be used relatively safely. You can change browsers, but your risk is the same with all of them -- nearly zero -- if your browser, OS, and all add-ons and plug-ins are fully patched.

However, if I pretended to be an end-user tricked into running a malicious executable (such as a fake anti-virus program), each browser allowed the system to be infected and compromised. End-users running on Windows Vista without elevated credentials would have prevented most malware infections from occurring, but even those users were readily exploited if they purposefully elevated themselves to install the rogue program.

## BROWSER SECURITY TIPS

Instead of accusing one browser of being weaker than another, real-world testing has revealed that users should pick a browser that has the security features and functionality they desire, and implement the following suggestions:

- Don't log on as admin or root when running an Internet browser (or use UAC on Windows Vista, SU on Linux, etc.).
- Make sure the browser, OS, and all add-ons and plug-ins are fully patched.
- Don't be tricked into running malicious code.
- If unexpectedly prompted to install third-party software while browsing a site, open another tab and download the requested software directly from the software vendor's Web site.
- Be careful about which add-ons and plug-ins you use. Many aren't secure, many are very insecure, and some are actually malware in disguise.

## BROWSER FINDINGS

As expected, each Web browser had its fair share of security advantages and disadvantages. All of the browsers reviewed here, save Google Chrome, have had years to mature in response to previous malicious attacks. All



of the browsers had SSL/TLS (Secure Sockets Layer/Transport Layer Security) support, anti-phishing filters, pop-up ad blocking, cross-site script (XSS) filtering, automated updates, private session browsing, and cookie handling. The following review summaries highlight their differences. Click the links to the full reviews for more detail. See also the table, "[Web browser security features](#)," comparing security features among all of the browsers.

**Google Chrome 1.0.** Google's first browser is a security paradox. It begins with the best browser security model, but then layers questionable decisions over a dearth of security features. It utilizes Windows Vista's new security features even better than the browser that came with Vista. JavaScript runs inside of a virtual machine environment, where it is further restricted.

Unfortunately, Chrome has almost no significant security granularity, and no separate security zones in which to place Web sites with different trust expectations. More disappointing, you cannot disable JavaScript at all. This is a huge security oversight, even if Google believes the browser can trap malicious JavaScript within the sandbox. Perhaps most troubling, Chrome has been plagued by relatively simple buffer overflow problems.

Chrome has the potential to be one of the most secure Internet browsers, but its initial showing only leaves significant questions. [Read the complete review.](#)

**Mozilla Firefox 3.12.** Mozilla's Firefox deserves the growing market share it has today. It is a battle-tested veteran with best-in-class cipher support, excellent add-on management, and growing enterprise features. Firefox has a fair amount of security granularity and is the only browser besides Internet Explorer to provide multiple security zones, although they are not easy to configure.

JavaScript can be disabled on a global basis, but it takes a separate add-on (called [NoScript](#)) to enable or disable it on a per-site basis. Using the About:security option in the URL bar allows the user to configure dozens of features and security settings, but the only enterprise deployment and management tools are offered by third parties. Firefox makes a good browser choice for anyone, especially for users who want to avoid the risk of native ActiveX support. [Read the complete review.](#)

**Microsoft Internet Explorer 8 beta 2.** Internet Explorer is the most frequently attacked browser in the world. Its popularity, complexity, and support of ActiveX controls gives it an elevated risk as compared to the rest of the competition. Still, it also has best-in-class enterprise support, superior security granularity, and multiple security zones in which to deploy Web sites with different trust requirements. It's the only browser with built-in parental controls and a granular add-on manager.

It is also the only browser with serious enterprise management features, providing more than 1,200 customizable settings across multiple security zones. For example, the U.S. government requires what is called FDCC (Federal Desktop Core Configuration) on all of its software, and FIPS (Federal Information Processing Standards) ciphers only. Tens of millions of PCs fall under these requirements. Only IE allows these policies to be enforced across all desktops. It is difficult to achieve with any of the other browsers.

IE 8 is bringing many new features to the table, including per-user and per-site control of ActiveX programs and other add-ons. Its improved base security model is second only to Google's Chrome, and nearly every security feature it has is mature and built for enterprise use. [Read the complete review.](#)

**Opera 9.63.** Opera is a solid browser that deserves more market share in the PC world. It has impressive security granularity, good anti-DoS handling, strict Extended Validation certificate handling, and many unique features. Its lack of market share means it hasn't been as tested as Internet Explorer and Firefox, but it has been involved in fighting many found vulnerabilities.

On the downside, Opera doesn't support DEP (Data Execution Prevention), ASLR (Address Space Layout Randomization), or ECC (Elliptical Curve Cryptography) ciphers. These deficiencies need to be corrected before its use can be more highly recommended. Even now, I invite readers to check out Opera. I think many people will be pleasantly surprised. [Read the complete review.](#)

**Apple Safari 3.2.1.** Apple's Safari browser has many good features, but lacks security granularity and zones. It has good pop-up blocking, good local password protection, and a surprisingly accurate anti-phishing filter.



Unfortunately, DEP is disabled, something that needs to be corrected. Safari has the weakest cipher support, failing to offer AES ciphers, 256-bit keys, or ECC ciphers.

Safari always automatically prompts the user before downloading files, and it prevents some high-risk files from being executed before downloading. Safari has good default cookie control. It is one of only two browsers in this review (the other is Chrome) to prevent all writes by third-party cookies by default, which is a nice privacy bonus. Although local password protection is strong, Safari had the weakest remote password handling of the bunch. Safari is a great-looking browser but a mixed bag with respect to security. [Read the complete review.](#)

You'll often hear how Internet Explorer is insecure because it is the most frequently attacked and the most complex browser. That's true, but it didn't allow any malware to silently install on the test systems. You'll also hear that the newer, less functional browsers, which haven't yet undergone the test of time, are more exploitable because they have less sophisticated security. That may be true as well, but again, none of the real-life malicious Web sites were able to exploit any of the browsers. In the end, the choice of which browser to use comes down to feature set (both security and non-security features), functionality, and the user's comfort level with the product.

I encourage you to check out the individual reviews to see what where the security strengths and weaknesses of each browser reviewed. (Note that my tests no longer reflect the most current versions of the browsers, and that some of their security features may have changed.) Most of all, however, don't forget the main lesson of these security reviews: A fully patched system prevented all silent attacks regardless of the browser. If you keep your browser, its add-ons, and the underlying operating system patched and up-to-date – and you're careful about which “helper” applications you download and run – the Internet will be a much safer place.

## **BROWSING BEHIND MANDATORY INTEGRITY CONTROLS**

Windows 7 and Windows Vista include a security construct called Mandatory Integrity Controls (MIC), which is similar to integrity functionality long available in the Linux and Unix worlds. In these later versions

of Windows, all security principals (users, computers, services, etc.) and objects (files, registry keys, folders, resources, etc.) are given MIC labels.

A subject of lesser integrity cannot modify (write or delete) an object of higher integrity, even if the normal NTFS permissions would otherwise allow it. Perhaps surprisingly, MICs take precedence over traditional file permissions, and it's critical that they do.

Security principals are assigned MICs in the form of SIDs (security identifiers) that are added to their access tokens during logon. Objects have MIC labels stored as part of their access controls (specifically within the System Access Control List portion, which is where the auditing attributes are stored as well). When a security principal (or a process on behalf of the security principal) attempts to access an object, both MICs are checked and their integrity evaluated.

Although many different integrity levels exist, Windows regularly uses six label values, including (proceeding from lowest to highest) Untrusted, Low, Medium, High, System, and Trusted Installer. Normal users have Medium integrity. The null/anonymous user is Untrusted. The default built-in Administrator and elevated members of the Administrators group have High integrity. The Windows system kernel and service files rank as System.

Most Web browsers run with Medium integrity by default. Add-on programs normally run as Medium or High. Internet Explorer in its default Protected Mode (enabled for all zones but Trusted Sites) runs with Low integrity for rendering processes, as does Google's Chrome. Although Internet Explorer was the first browser to support MICs, Chrome actually utilizes them to a greater extent ([see the Chrome review](#)). Along with preventing lesser integrity writes, Chrome prevents lesser integrity reads as well.

The purpose behind integrity controls, of course, is to give Windows another layer of defense against malicious hackers. For example, if a buffer overflow is able to crash Internet Explorer (and not a third-party add-on or toolbar), the resulting malicious process will often end up with Low integrity and be unable to modify Windows system files. This is the primary reason so many Internet Explorer exploits have resulted in an “Important” severity rating for Vista but a higher “Critical” rating for Windows XP.

Every Web browser should make use of the integrity controls in Windows 7 and Windows Vista to the fullest



extent possible. Their use significantly improves protection for the end user. Browsers running in Low integrity (e.g. Chrome, Internet Explorer) offer additional protections that the others don't, but should.

## SECRETS OF SECURE BROWSER CONNECTIONS

Although most users don't know it, their Web browser plays a key part in determining the strength of the ciphers used between their client and an HTTPS-protected Web site. Encryption ciphers used in the SSL/TLS (Secure Sockets Layer/Transport Layer Security) negotiations can range from very strong to weak, and involve asymmetric ciphers, symmetric ciphers, key exchange algorithms, and hash functions.

SSL has been replaced by TLS 1.0 as the current HTTPS standard. It is possible in many browsers to select which SSL and TLS versions are enabled. Any browser you use should support TLS, and offer it by default to HTTPS-protected Web sites. Most browsers still support SSL v.3.0, which is next closest in strength to TLS. Many browsers still support SSL v.2.0, but many will have it disabled by default. SSL v.1.0 is considered insecure today, although some browsers will still use it.

## CIPHERS

Common TLS/SSL symmetric ciphers, in order of strongest to weakest (generally), include Advanced Encryption Standard (AES), Triple DES (3DES), RC4, Data Encryption Standard (DES), and RC2. Every browser today should be offering AES as its default symmetric protocol, followed by 3DES as a backup. The other protocols should only be used for legacy compatibility or offered last.

Common TLS/SSL asymmetric ciphers, in order of strongest to weakest (generally), include Elliptical Curve Cryptography (ECC), Rivest Shamir Adleman (RSA), and Diffie Hellman (DH, or DHE for key exchange). ECC is the newly crowned standard for asymmetric ciphers these days, and included in the U.S. Government's Federal Processing Information Standards (FIPS) as part of what is called Suite B. A browser must support Suite B to be considered for use by the U.S. Government. Browsers should offer ECC as the first asymmetric cipher, followed by RSA or DH.

## HASH FUNCTIONS

Common cryptographic hash functions include, in order

of strongest to weakest (generally), SHA-512, SHA-384, SHA-256, SHA-1, and MD5. MD5 and SHA-1, by far the most popular hashes used, have been shown to have some moderate cryptographic weaknesses, and as such, crypto experts recommend stronger hashes. Of the two, MD5 has been demonstrated to have larger cryptographic weaknesses. Recently, it was shown that digital certificates signed using MD5 hashes cannot be relied upon. Users should avoid cryptography using MD5, and strive to use SHA-2 (the family of SHA-256, SHA-384, SHA-512) whenever possible.

## KEY SIZES

SSL/TLS symmetric key sizes often range from 40-bit (the old SSL standard) to 512-bit (very strong). Symmetric key sizes of 128-bit to 256-bit are considered secure for most normal security operations. 256-bit keys are just now becoming the standard, although 128-bit keys are still the most popular.

In general, longer key sizes are stronger within a particular cipher. For example, a 256-bit AES key is stronger than a 128-bit AES key. However, you can't always use key size as a strength measurement between cipher families. For example, 384-bit ECC is considered stronger than 1024-bit Diffie-Hellman. Plus, you can have a really horrible cipher with a really long key size and still have poor protection. As a matter of fact, users should be wary of newly announced ciphers from questionable sources that claim ultra long key sizes (e.g. 1 million bits, etc.). A good cipher doesn't need an ultra long key size. If the cipher algorithm is good, smaller key sizes can be used and the cipher will remain strong.

## BROWSER CIPHER ORDER

When a browser first connects to a SSL/TLS protected Web site, the first packet in the SSL handshake includes the browser's preferred cipher order", including all the ciphers the browser currently supports. Both the client and the Web site must agree upon which ciphers to use before they continue. With any luck, the Web site will pick the strongest cipher the client supports.

By offering the strongest cipher first, the browser increases the likelihood that a Web server will pick it, if it supports it. Using stronger cipher orders shows a browser vendor's commitment to cipher strength. Still, it is not





unusual to see a browser vendor support very strong ciphers but offer up weaker, more popular ciphers first. This could potentially speed up SSL/TLS negotiations.

## THE BROWSERS COMPARED

Browsers in this review run the gamut in cipher support.

Firefox (v.3.12) has the strongest first cipher showing (TLS, ECC, AES, 256-bit key) followed by Opera (v.9.63). Firefox also has strong defaults, and 34 total ciphers to choose from.

Opera is impressive because it offers 256-bit symmetric ciphers for the first five suggestions (TLS, RSA, AES being the first). However, Opera doesn't offer ECC support at all, which means that Chrome (v1.0) and Internet Explorer (v.8 beta 2), which do offer ECC, could easily be considered tied for second in cipher support if more than first cipher offered were considered.

Both Chrome and Internet Explorer offer TLS, RSA, AES with a 128-bit key first, and with a 256-bit key second. In both cases, ECC isn't offered until fifth. Still, Safari runs away with last place with weak first offerings (TLS, RSA,

RC4, 128-bit key is offered first and second), frequent MD5 offerings, and no support for ECC, AES, or 256-bit keys.

## BROWSERS VS. XSS ATTACKS

The Web is full of cross-site scripting (XSS) attack warnings. Essentially, XSS refers to the injection of malicious JavaScript into a legitimate Web page, where it can then be executed in the browsers of innocent visitors. A Web site is susceptible to XSS if it allows users to upload content to be shared with others and does not thoroughly inspect that content to remove potentially malicious scripts.

Take, for example, a Web site that hosts a user contributable blog. Perhaps all the Web developer wanted was for users to be able to upload plain text, and never considered that this would also allow scripting. Because of this oversight, the developer never thinks to filter the content. Bad mistake.

A common test script used to determine whether a Web site is vulnerable to XSS is:

```
<SCRIPT>alert("XSS is possible");</SCRIPT>
```

If you can upload that content to the Web site, and you can see the alert when the page is viewed, then the Web site is XSS exploitable. An excellent tutorial on XSS issues is available on [the Open Web Application Security Project Web site](#).

What's the big deal with executing a few unexpected JavaScript commands? After all, the JavaScript execution can only do what the user can do in the user's own security context, right?

Unfortunately, a lot of bad things can be done in the user's context, especially when the majority of Windows users are running as Administrators. XSS attacks have been highly successful at reaching outside of the browser to steal confidential information the user would otherwise not want to share. In some cases they can cause buffer overflows and even complete system compromise.

Once during my professional penetration testing days, my team was hired to break into a cable company through its own cable set top devices. The set top boxes were completely HTML-enabled and allowed tremendously fine-grained control over the user's experience. They even included a host-based firewall.


A quick check, using the test script string noted above,

Browser	Preferred SSL/TLS cipher	Related notes
<b>Firefox 3.12</b>	TLS, ECC, AES, 256-bit key	Strongest in review, strong defaults, 34 total ciphers to choose from
<b>Opera 9.63</b>	TLS, RSA, AES, 256 bit key	First five ciphers offer 256-bit keys, no ECC, 17 ciphers offered by default, but another 4 weaker SSL 3.0 ciphers can be enabled
<b>Chrome 1.0</b>	TLS, RSA, AES, 128-bit key	TLS, RSA, AES 256-bit offered second, ECC offered fifth; 12 ciphers offered
<b>Internet Explorer 8 beta 2</b>	TLS, RSA, AES, 128-bit key	TLS, RSA AES 256-bit key offered second, ECC offered fifth; 12 ciphers offered
<b>Safari 3.2.1</b>	TLS, RSA, RC4, 128-bit key offered first and second	ECC, AES, and 256-bit keys are never offered, SSL2 offered; 17 ciphers offered



showed that the firewall's log file was XSS exploitable. I created a script and "injected it" simply by attempting to attack the device remotely; although my attack was unsuccessful, the set top device duly recorded the script in its log file. I then called the cable company's tech support company and complained that my set top box was being attacked, and asked kindly if tech support could confirm it by reviewing the log files.

Within a few seconds I had the tech support's password and shadow files (i.e. their Linux-based password storage files) in hand. The script I had injected instructed the tech support's PC to send their password files to my e-mail account. My co-worker and I quickly sifted through the resulting password files, found everything we needed to know, and quickly took over the cable company's entire network. We had been hired to break into the company's set top box to see what mischief we could cause there, and within a few hours we owned the corporation's global network. That's the power of XSS attacks and why you must take them seriously.

All of the reviewed browsers included at least some built-in XSS mitigations to prevent malicious exploitation to varying degrees. And all passed the common XSS tests available on the Internet used in this review. That's the good news. Avoid any browser whose vendor claims that anti-XSS defenses are the sole domain of Web site owners. Also keep in mind that it may be a good idea to disable JavaScript (if your browser allows it) on Internet sites when a new XSS or JavaScript-enabled vulnerability is announced. And if you come across a Web site that appears vulnerable to XSS attacks, notify the content owner. Often they are unaware that their user friendly site could be the host of unintended maliciousness. 

---

*Roger A. Grimes (CPA, CISSP) is senior contributing editor and Security Advisor columnist at InfoWorld. A 23-year Windows security consultant, instructor, and author, he currently works full-time for Microsoft, where he serves as principal security architect for the Microsoft InfoSec ACE Team.*