JEREMIAH GROSSMAN

top 10 Web hacking techniques: "what's possible, not probable"

Jeremiah Grossman, founder and CTO of WhiteHat Security, is a world-renowned Web security expert. A co-founder of the Web Application Security Consortium (WASC), he was named to *InfoWorld*'s Top 25 CTOs in 2007 and is often quoted in major publications such as *SC Magazine*, *Forbes*, and *USA Today*. He has authored dozens of articles and white papers, is credited with the discovery of many cutting-edge attack and defensive techniques, and is a co-author of *XSS Attacks: Cross Site Scripting Exploits and Defense*. Grossman is also an influential blogger who offers insight and encourages open dialogue regarding research and vulnerability trends. Prior to WhiteHat, Grossman was an information security officer at Yahoo!, responsible for performing security reviews on the company's hundreds of Web sites.

jeremiah@whitehatsec.com

NEW ATTACK TECHNIQUES PROVIDE keen insights into the state of the security of the Web. Web client attack techniques impact online businesses, reveal what may become the next popular exploit technique, and may affect anyone who uses a Web browser. In this article, I cover the top 10 Web hacking techniques, a selection chosen by a panel of security experts from a field of 70 candidates [1].

Sharing technical details of these hacking techniques isn't meant to give malicious hackers a set of instructions, but to level the playing field for the good guys. Without this information, defenders would be unfairly handicapped against a determined criminal element who targets the Web as their primary attack vector. Notification of vendors by researchers also provides vendors with a chance to patch their software before it can be exploited.

It is unclear which of these, if any, will become a widely used method of attack. What we do know is that some have already been used against us. The following hacking techniques were ranked by a panel of four widely recognized security experts (Rich Mogull, Chris Hoff, H.D. Moore, and Jeff Forristal) based on their novelty, impact, and pervasiveness. With that I give you the Top 10 Web Hacking Techniques of 2008!

1. GIFAR by Billy Rios, Nathan McFeters, Rob Carter, and John Heasman [2]

A GIFAR is the concatenation of a GIF image and Java Archive (JAR) containing a potentially malicious Java Applet. Many Web sites take ownership of user-supplied content (e.g., image uploads) after parsing the bytestream beginning to end and ignoring trailing "garbage" data. In the case of GIFAR the trailing garbage data is a compressed Java Applet, a JAR, which is essentially a zip file parsed bottom up. When Web sites take ownership of a GIFAR because it "looks" like a GIF, the attached Java archive may execute arbitrary applet code in the victim's browser under the context of the domain from where it was loaded. This results in a same-origin policy violation, similar in scope to that of a persistent cross-scripting vulnerability. Furthermore, the GIF portion of GIFAR can be substituted for any file type the Web site will accept and parse in a top-down fashion (i.e., JPG, DOC, MP3, etc.).

2. Breaking Google Gears' Cross-Origin Communication Model by Yair Amit [3]

Google Gears is a browser extension that allows developers to create rich and responsive Web applications. Of the many available features, Google Gears offers developers cross-origin communication capabilities, making it much easier to implement mash-ups, for example. Under some circumstances the cross-origin communication security model of Google Gears may be bypassed by an attack that inserts malicious code. If an attacker can upload arbitrary "worker" code (the JavaScript code that can access Gears capabilities) to target a Web site, the attacker can issue malicious commands under that domain. This worker code is likely to pass input security controls, as it lacks suspicious tokens such as <script> tags.

3. Safari Carpet Bomb by Nitesh Dhanjani [4]

The Safari Carpet Bomb attack allows a malicious Web site to litter the user's desktop on Microsoft Windows or the user's "Downloads" directory on OS X with arbitrary files or malware. Unless patched, when the Safari browser is served a file with a content type that cannot be rendered by the browser, it automatically downloads it to the default download location without notifying or asking the user. This "carpet bomb" attack may trick users into clicking on the malicious files by mistake or through curiosity. Safari Carpet Bomb has the distinction of bringing the term "blended threat" into the security vernacular, because if you are able to litter user's machines with arbitrary files, you can further the impact and affect other applications that trust content on the local file system.

4. Clickjacking/Videojacking by Jeremiah Grossman and Robert Hansen [5]

Think of any button (image, link, form, etc.) on any Web site that can appear between the Web browser walls. This includes wire transfer forms from bank sites, DSL router buttons, Digg buttons, CPC advertising banners, Netflix queue, Facebook friend requests, and so on. Next consider that an attacker can invisibly hover these buttons below the user's mouse using iframe tags and CSS opacity functionality, so that when a user clicks on something they visually see, they're actually clicking on something the attacker wants them to—you now have clickjacking. We also demonstrated that clickjacking can be used to trick users into enabling a Web cam and microphone through a Flash movie to enable remote surveillance. If you haven't done so already, I strongly suggest you upgrade to Flash version 10 or later or at least cover up the camera with a Post-It note. Finally, cross-site request forgery defenses using one-time tokens (nonces) can also be bypassed using clickjacking.

5. A Different Opera by Stefano Di Paola [6]

Until it was patched, the Opera Web browser itself was vulnerable to a cross-site scripting vulnerability in the History Search page, where JavaScript execution occurred under the opera:* context. Using iframe tags and a cross-site request forgery, this provided a malicious attacker with the ability to modify browser settings under opera:config, specifically the "mailto" preference. Updating the mailto preference to an arbitrary value could enable the arbitrary execution of operating system commands.

6. Abusing HTML 5 Structured Client-Side Storage by Alberto Trivero [7]

HTML 5 has introduced three powerful new ways to store significant amounts of data on the client's PC through the browser. This allows storage of much more data than standard cookies, in Session Storage, Local Storage, and Database Storage. If a Web application using this kind of client-side storage is vulnerable to cross-site scripting, attackers can use their payload to read or modify the content of known storage keys on the computer's victim. If the Web application loads data or code from the local storage, this could also be a powerful method to inject malicious code that will be executed every time the Web application requests it.

7. Cross-Domain Leaks of Site Logins via Authenticated CSS by Chris Evans and Michal Zalewski [8]

Web browser vendors take great pains to ensure that their same-origin policy prevents code on one Web site from obtaining details, such as authenticated content or session cookie data, from another Web site. Violations of the same-origin policy, such as the ability to determine if a user is actively logged on to an arbitrary Web site (e.g., a social network), has serious security and privacy implications. One way this can be achieved is through the inline inclusion of authenticated Cascading Style Sheets on off-domain locations by a malicious Web page. The malicious Web page checks to see if unique CSS properties have been loaded by the off-domain Web page using standard JavaScript APIs. If so, the user is logged in—a simple Boolean result. Similarly, this same attack can be performed with content that only appears in authenticated sessions, including images and JavaScript files.

8. Tunneling TCP over HTTP over SQL Injection by Glenn Wilkinson, Marco Slaviero, and Haroon Meer [9]

The common Web infrastructure is designed using a multi-tier architecture. A client connects to the server (port 80/443), which connects to back-end databases and applications to generate dynamic content. Remote clients may not directly connect to the back-end systems, where the crown jewels are located, as the server can, and certainly cannot communicate with them over arbitrary protocols and ports—that is, unless the server has a SQL injection vulnerability. In this technique, squeeza, a tool for exploiting SQL injection, is used to upload reDuh to the vulnerable server as a JSP, PHP, or ASP file. reDuh, when executed as a Web application on the vulnerable server, creates a TCP tunnel through validly formed HTTP requests using a client-server model. reDuh gives an attacker access to the server behind the first-layer firewall, which then acts as a relay to communicate with any reachable back-end system.

9. ActiveX Repurposing by Haroon Meer [10]

Resident or latent ActiveX controls, including those used to access SSL VPNs, can be abused by a malicious attacker. In this technique, a particular ActiveX control included the features to update itself if the server informed it of a new software version. By simply instantiating the control and passing it a higher build number and a URL path to a downloadable file, it would cause the client to download a possibly malicious file. Before loading the control, Internet Explorer would first check the downloaded file to see if it was properly signed. If it was not, then the file would not be executed. However, the file would still download to a predictable location on the local file system, where it would remain. Upon first malicious instantiation, an attacker would force the control to download a mock configuration file it supported. The second instantiation would call the control and point to the previously downloaded configuration file, which could contain arbitrary operating system commands, including an uninstall method.

10. Flash Parameter Injection by Yuval Baror, Ayal Yogev, and Adi Sharaban [11]

Flash parameter injection introduces a new way to inject values into global parameters in Flash movies while the movie is embedded in its original HTML environment. These injected parameters can grant the attacker full control over the page DOM, as well as control over other objects within the Flash movie. This can lead to more elaborate attacks which take advantage of the interaction between the Flash movie and the HTML page in which it is embedded. There are several different FPI variants, and most include tricking the server into sending back a page where user input is interpreted as Flash parameters. This allows an attacker to inject malicious global parameters to the Flash movie and exploit Flash-specific vulnerabilities. When an attacker is able to access and control global Flash parameters, he can achieve attacks such as cross-site scripting through Flash, cross-site flashing, and changing the flow of the Flash video.

Conclusion

There is a difference between what is possible and what is probable, something we often lose sight of in the world of information security. For example, a vulnerability represents a weakness an intruder may exploit in an asset by way of a particular attack technique, such as those described above. Obviously, a vulnerability's mere existence does not necessarily mean it will be exploited or indicate by whom or to what extent. Some vulnerabilities are more difficult to exploit than others and therefore attract different attackers. Often a particular attack technique will only become widely used maliciously years after initial discovery, similarly to SQL injection. This is why we are exploring them now.

What we do know is that attack techniques tend to only be taken seriously after they are both well understood and respected. We can assist with understanding through awareness efforts but, unfortunately, historically respect is gained through mass exploitation.

REFERENCES

[1] Top Ten Techniques blog entry: http://jeremiahgrossman.blogspot.com/2009/02/top-ten-web-hacking-techniques-of-2008.html.

[2] GIFAR technique: http://xs-sniper.com/blog/2008/12/17/sun-fixes-gifars/.

[3] Breaking Google Gears' cross-origin communication protection: http://blog.watchfire.com/wfblog/2008/12/breaking-google-gears-cross-origin-communication-model.html.

[4] Safari Carpet Bombing: http://www.dhanjani.com/blog/2008/05/safari-carpet-b.html.

[5] Clickjacking: http://www.sectheory.com/clickjacking.htm.

[6] Opera History attack: http://seclists.org/fulldisclosure/2008/Oct/0401.html.

[7] Abusing HTML 5 Structured Client-Side Storage: http://trivero.secdiscover.com/html5whitepaper.pdf.

[8] Cross-domain leaks of site logins via Authenticated CSS: http://scarybeastsecurity.blogspot.com/2008/08/cross-domain-leaks-of-site-logins.html.

[9] Tunneling TCP over HTTP over SQL Injection: http://www.sensepost.com/research/reDuh/.

[10] ActiveX Repurposing: http://www.sensepost.com/blog/2237.html.

[11] Flash Parameter Injection: http://blog.watchfire.com/wfblog/2008/10/flash-parameter.html.