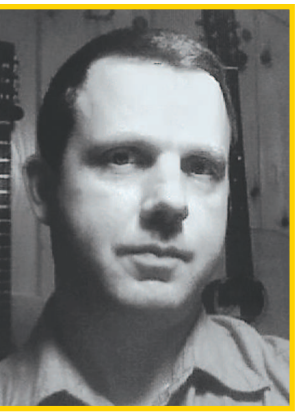


Connect

Fiddler with Watcher: Passive security auditor

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

Prerequisites

Windows OS
Fiddler

Similar Projects

RatProxy, ProxMon, Paros Proxy, Burp Suite

Chris Weber's Watcher[1] is a Fiddler[2] add-on that provides a runtime, passive analysis tool best used to detect web application security issues and operational configuration issues. For those of you who are unfamiliar with Fiddler, it's a free web debugging proxy which allows you to capture all HTTP(S) traffic from virtually any application and inspect same, set breakpoints, and "fiddle" with incoming or outgoing data. As a Fiddler enhancement, Watcher is ideally positioned to allow penetration testers immediate detection for vulnerability hotspots, but developers can use it to review deployed applications, while auditors can make use of Watcher to review for PCI compliance as well as adherence to OWASP and SDL[3] standards. Chris let me know that Watcher will be amongst tools announced as SDL recommended tooling and that support will be added for Team Foundation Server (TFS) so that bugs can be filed directly from Watcher to a TFS server, a platform that allows source control, data collection, reporting, and project tracking during collaborative software development projects.[4] Chris also stated that a complimentary XSS auto-testing tool is expected late 2009/early 2010.

Watcher works passively at runtime, requiring you to utilize a browser and navigate a web application as an end user. Watcher and Fiddler work with any browser; ultimately, all HTTP(S) traffic can be proxied through Fiddler, and thus Watcher. Watcher will allow discovery of issues related to cross-domain mashups, user-controlled HTML (potential XSS), open redirects, insecure handling of cookies, Unicode, and other prospective vulnerabilities such as:

- Cross-domain stylesheet and JavaScript references
- User-controllable cross-domain references
- Cross-domain form POSTs
- Potential XSS with user-controllable HTML attribute values such as href, form action, etc.
- Cookies that don't set the *HTTPOnly* flag
- Cookies set over SSL that don't include the *secure* flag
- Loosely-scoped cookies
- Open redirects which can be abused by spammers and phishers
- Insecure Flash object parameters useful for cross-site scripting
- Insecure Flash crossdomain.xml
- Insecure Silverlight clientaccesspolicy.xml
- Charset declarations which could introduce vulnerability (non-UTF-8)
- User-controllable charset declarations
- Charset mismatches between HTTP and HTML
- Dangerous context-switching between HTTP and HTTPS
- HTTP pages insecurely loading HTTPS forms
- Insufficient use of cache-control headers when private data is concerned (e.g., no-store)
- Potential HTTP referer leaks of sensitive user-information
- Information disclosure from URL parameters
- Source code comments worth a closer look
- JavaScript that uses eval functions
- Insecure authentication protocols like Digest and Basic
- SSL certificate validation errors
- SSL insecure protocol issues (allowing SSL v2)
- Unicode ill-formed UTF-8 byte sequences
- SharePoint servers with insecure configurations

Included amongst Watcher's 35+ checks is coverage for Microsoft SDL requirements and OWASP recommendations, aiming to discover obvious issues as well as hidden or overlooked problems.

Installing and running Fiddler and Watcher

Setup is as easy as first installing and running Fiddler: download and execute *Fiddler2Setup.exe* then start it from the Windows *Start* menu or click *Fiddler2* in the Internet Explorer *Tools* menu. Follow this effort by launching the Watcher installer acquired from the CodePlex site, or manually drop the Watcher DLL's in Fiddler's scripts folder. With Fiddler running click the *Watcher* tab and choose *Enable* in the *Configuration* tab; you can also specify domains here to enhance cross-domain issue detection. The *Checks* tab provides a view to the 37 checks as I write this.

1 http://websecuritytool.codeplex.com.

2 http://www.fiddler2.com/fiddler2.

3 http://blogs.msdn.com/sdl/archive/2009/04/16/watcher-a-new-web-security-testing-tool.aspx.

4 http://msdn.microsoft.com/en-us/teamsystem/default.aspx.

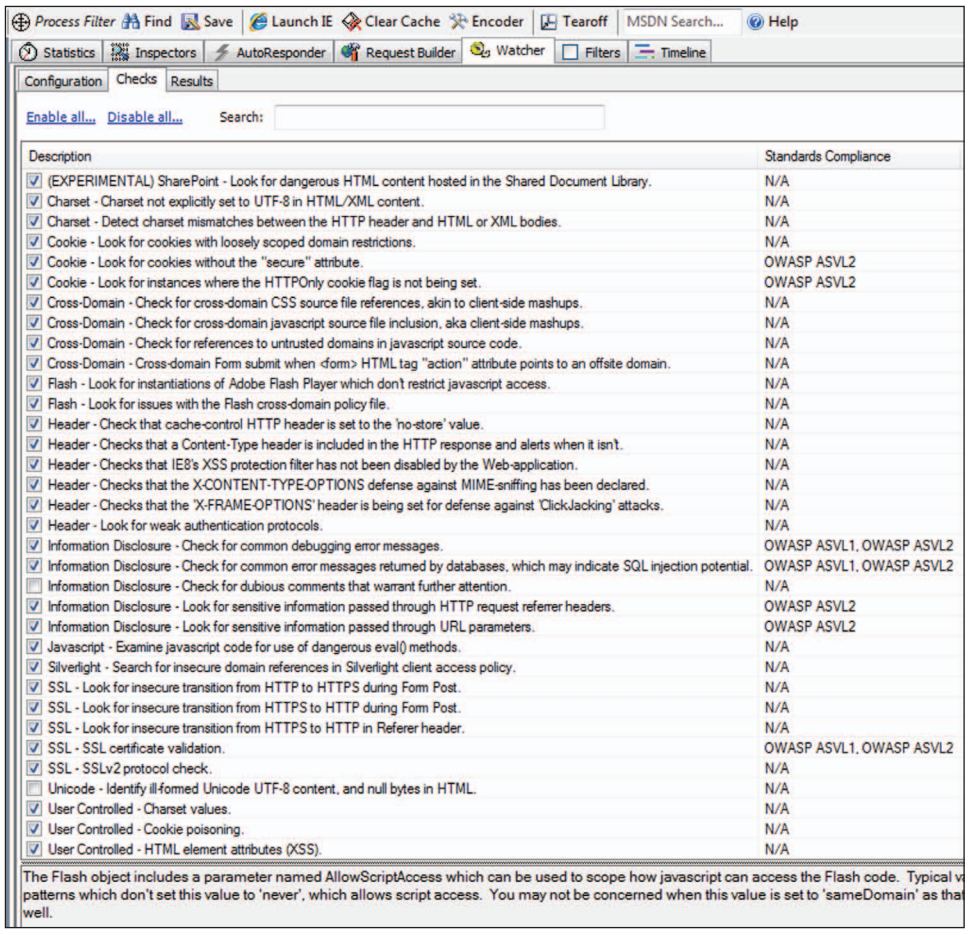


Figure 1 – Watcher's long check list

Switch to the *Results* tab and you'll see Watcher findings scroll by as it passively monitors all HTTP(S) traffic.

Watcher at work

I get a huge kick out of items certain Watcher checks uncover.

The "*Information Disclosure – Check for dubious comments that warrant further attention*" check warrants further commentary. ;-)

I was querying for sites that might help track down faulty Flash to investigate with Watcher, and HP's SWFScan blog post was returned as one of the search results. Ironically, utility JavaScript at HP's software security blog trigger a Watcher result for dubious comments as seen in Figure 1.

Granted, a bug where XMLHttpRequest sends an incorrect content-length header is not all that remarkable, but Watcher's ability to quickly grep it out of HP's `prototype.js` is slick at a minimum.

Watcher spots almost all XSS prospects; you still have to do the work, but it will certainly point out where the reviewer should get busy, including potentially vulnerable parameters/variables.

Another one of my favorites is the Cookie – Look for instances where the HTTPOnly cookie is not being set. I can't think of a reason not to set the HTTPOnly flag, and Watcher will quickly point out sites that don't. Yes, setting the HTTPOnly flag is optional but doing so ensures that the cookie cannot be accessed through client side script (think cross-site scripting (XSS)) assuming browser support for the flag.[5] Thus, even if an XSS flaw exists, and a user succumbs to an exploit attempt, the browser (typically IE and FF) will not reveal the cookie to a third party.[6] To quote Jordan Wiens, "No cookie for you!"

The same above mentioned HP software security blog does not set the HTTPOnly flag. Not the end of the world, but an interesting decision none the less. Note that because many cookies are non-sensitive and don't justify an HTTPOnly flag, Watcher includes a filter in the configuration option where you can specify the specific cookie you're concerned with. For example, where a site might offer as many as 20 cookies, but only 3 are auth type cookies, you can filter accordingly.

Open redirects are problematic[7] to say the least. Simply, they make it even easier for phishers to exploit their victims as

5 http://msdn.microsoft.com/en-us/library/ms533046%28VS.85%29.aspx.

6 http://www.owasp.org/index.php/HTTPOnly.

7 http://cwe.mitre.org/data/definitions/601.html.

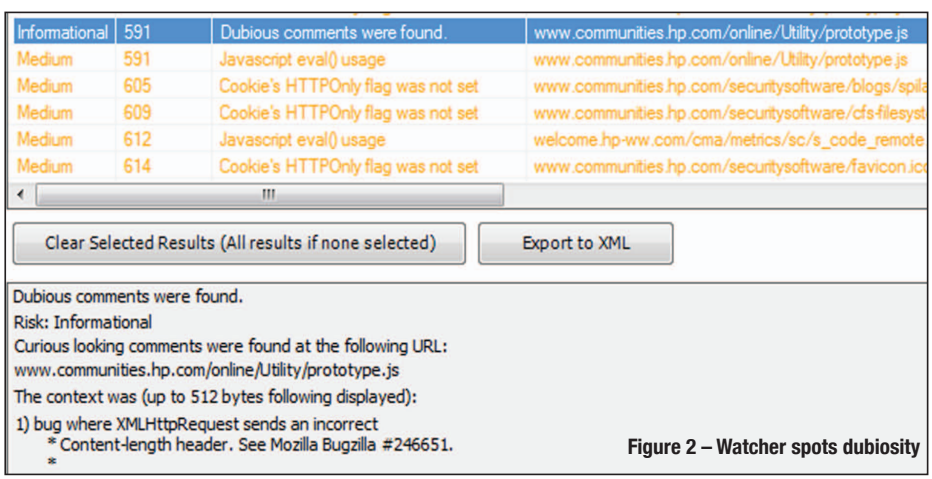
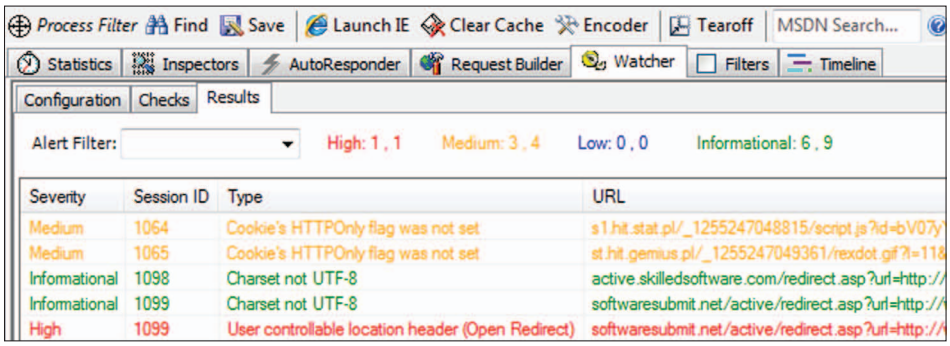


Figure 2 – Watcher spots dubiosity



URLs appear to originate from known good sites. If you're ever bored and want to see how widely available open redirects are, just try the following Google search: *inurl:"redirect.asp?url="*.

More irony discovered when a SEO business such as Software Submit is revealed to allow open redirection when visiting their site with Watcher. This shouldn't be surprising given their business model, but as seen in Figure 3, it's still bad practice.

Figure 3 – Watcher spots an Open Redirect

My final pet peeve spotted by Watcher pertains to bad Flash practices. In particular, if a page specifies a potentially insecure *allowScriptAccess* value[8] when loading a flash SWF file, or a crossdomain.xml policy file is too permissive,[9] an attacker is afforded additional inroads to exploit opportunity.

Here's what your crossdomain.xml policy file should not look like:

<cross-domain-policy>

<allow-access-from domain="" secure="false"/>*

<allow-http-request-headers-from domain="" headers="*" secure="false"/>*

</cross-domain-policy>

If you configure an origin-domain in Watcher's *Configuration* tab, Watcher will give you a heads up when crossdomain.xml is too permissive. The Watcher development team is updating Watcher so that even if an origin-domain is not specified, Watcher will default to using the response's domain. This should ensure that Watcher identifies all poorly implemented cross-domain policies.

In conclusion

Watcher is so simple to make use of in concert with Fiddler. If you spend any time conducting any sort of web-specific analysis, you'll find Watcher extremely useful and easy to utilize; utilizing the results as part of preliminary reconnaissance will serve you well.

Cheers…until next month.

Acknowledgment

—Chris Weber, Casaba Security, Watcher developer

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.

8 http://kb2.adobe.com/cps/164/tn_16494.html.

9 http://jeremiahgrossman.blogspot.com/2008/05/crossdomainxml-invites-cross-site.html.