YGN ETHICAL HACKER GROUP

Article

A Most-Neglected Fact about Cross Site Request Forgery

By

Aung Khant

Last updated: August 15, 2010

A Most-Neglected Fact about Cross Site Request Forgery (CSRF)

For those who don't know about *Cross Site Request Forgery* (CSRF/XSRF), a generalized meaning is that CSRF is a web application vulnerability where a malicious web site can make legitimate requests to a vulnerable web site under the disguise of a logged-in user without that user's knowledge. If a user can send email to his friend, then a malicious web site can do the same. This vulnerability has been rated as one of OWASP (Open Web Application Security Project) Top 10 vulnerabilities – A5, and Common Weakness Enumeration (CWE – 352).

When a CSRF attack or vulnerability is mentioned, people tend to think of a phishing vector (so called "a malicious web site") for it to be successful. Thus the threat of CSRF seems to be small as more and more people have been aware of phishing than the past. As a result, the CSRF protection is often overlooked by web application developers. In fact, it can be executed by several other vectors in addition to a malicious web site that a victim has visited via a phishing email.

Scenario 1 – A Stealth Backdoor

Consider a factitious Gecko browser add-on or a simple Greasemonkey script backdoor. Assume a user accidently installed a cool add-on that contains a backdoor. Similarly, several Microsoft Windows based browsers/applications embed Microsoft Internet Explorer and inherits its settings/user profiles. Assume a user accidently installed a cool application that has a backdoor with IE embedded.

A backdoor may have a fixed post data where it can inject them when a victim user has logged in to a particular web application. Rather than stealing username and password and submitting them to its master, that backdoor records every high-profile web site that a victim visits. At random idle time while a user's logged-in session is valid, it will do malicious requests such as modifying account settings, adding a backdoor user…etc. A backdoor can own everything but trying to own everything makes a vigilant user knock his sense of "something wrong". By utilizing a CSRF flaw, a backdoor can maintain its stealth movement because it does not have to connect to its command and control server while its master, an attacker, can watch its successful CSRF injection at their own pace without interacting with a victim.

Scenario 2 – A Third-party Widget

Consider a web site has several third-party widgets which let the site use their service by use of embedded JavaScript files. Assume a malicious hacker had compromised a widget host. He embedded a stealthy backdoor JavaScript which can detect whether a user has logged in through the analysis of current cookie values. Upon detecting a

session cookie, it would perform a CSRF on the vulnerable host, attacking its logged-in users at mass.

Scenario 3 – A Multi-user application

Today's web applications are multi-user based, consisting of so-called 'members'. Every relatively secure web application prevents users from injecting JavaScript in places such as form posts, comments, email body and so on. However, such multi-user applications have to give users freedom to do such as embedding arbitrary images, hyperlinks and so on. A malicious user can embed image with src tag set to a CSRF payload by posting an interesting topic about a vulnerable web site. If a request is a type of POST, then he can prepare such POST data in a flash file combined with an interesting movie snapshot or stuffs like that. Similarly, an attacker can perform CSRF to users of vulnerable web mail service where HTML email option is allowed. Most technical savvy users would think an image in an email message can only be used to do user tracking which in fact an image can add new email address to their contact list or delete their messages and empty the trash.

Scenario 4 – A Trusted Application

The scenario two is exploiting the trust of a victim application on its third-party widget.

Consider a similar scenario. A victim user was a member of a feed reader web application which accepted his desired feeds, notified him of new feed entries, and allowed him to view them in its member area. The application directly displayed external entities such as images, flash, Silverlight in the feed display views. A malicious user posted a flash file embedded with CSRF payloads, which caused the victim user's password reset to the attacker's arbitrary value.

Scenario 5 – A Supporting Attack Vector

People think that the CSRF is just doing undesired cross site requests on taking advantage of a logged-in user. Attackers can leverage this more.

Consider a member-only web application with terrible flaws like persistent Cross Site Scripting (XSS) and SQL Injection. Let's say only privileged people were allowed to be members. So, an attacker had a hard time to successfully exploit such flaws against his target site.

After coming up with an idea of combined attacks vector, he prepared a CSRF with different payload vectors linked with XSS and SQL Injection. At a time, he could successfully do CSRF to one of the application users. This CSRF saved persistent XSS payload in the victim's profile pages. The attacker then launched an XSS proxy

that allowed him to read the entire victim's private pages. It also let him do successful SQL Injection through that proxy, which provided him useful information from the database. The web server log showed the intrusion was come from the victim's IP and the log filled with his traces not the attacker's.

The CSRF was the main supporting attack vector for the attacker to achieve his mission one by one.

Conclusion

As seen in the above scenarios, the CSRF is no longer an issue that can be neglected in today's web application arena. The CSRF armed with the HPP (HTTP Parameter Pollution) attack vector can even bypass web browser-based filter, web firewall, and defense employed in applications. Tools used to identify and prepared CSRF are readily available. The impact of the CSRF grows bigger.

References:

[1] OWASP 2010 Top 10 – A 5 | http://www.owasp.org/index.php/Top_10_2010-A5-Cross-Site_Request_Forgery_%28CSRF%29

[2] CWE – 352 | http://cwe.mitre.org/data/definitions/352.html

[3] Backdoor in Web Sniffer Add-on | http://news.netcraft.com/archives/2010/07/15/firefox-security-test-add-on-was-backdoored.html

[4] HTTP Parameter Pollution | http://www.owasp.org/images/b/ba/AppsecEU09_CarettoniDiPaola_v0.8.pdf

[5] XSS Tool: Shell of the Future | http://www.andlabs.org/tools/sotf/sotf.html

[6] XSS Tool: Imposter | http://www.andlabs.org/tools/imposter/Imposter.html

[7] XSS Tool: Beef | http://www.bindshell.net/tools/beef/

[8] CSRF Tester | http://www.owasp.org/index.php/Category:OWASP_CSRFTester_Project

[9] Pinata | http://code.google.com/p/pinata-csrf-tool/

[10] Fiddler XSRF Inspector | http://sourceforge.net/projects/xsrfinspector/