Preparing for the Cross Site Request Forgery Defense

Presentation Handout and Reference
By Chuck Willis – chuck.willis@mandiant.com

Presented at Black Hat Briefings DC 2008 on February 20, 2008
Slides available at www.blackhat.com.

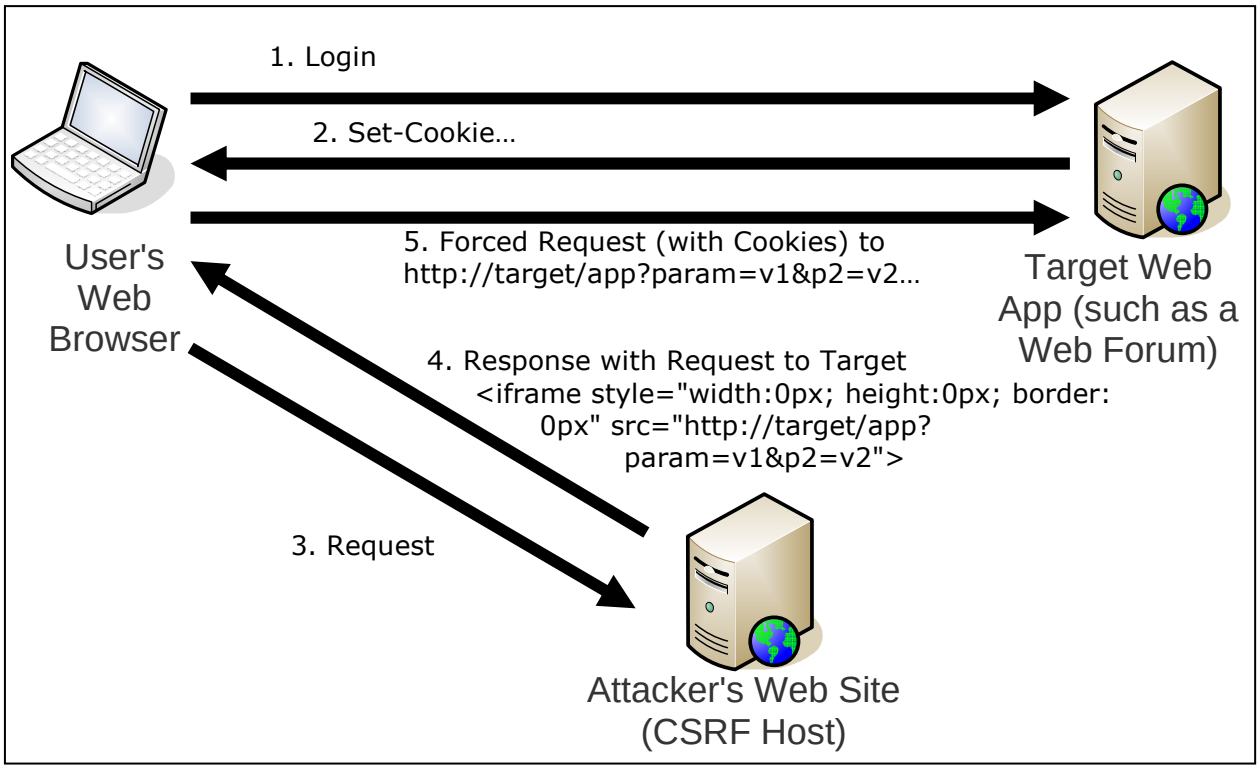
Abstract:

Computer Forensic cases often involve an analysis of a user's web browser's cache and history in order to reconstruct actions that a user took online. Corporations also routinely monitor Internet activity of users and illicit or inappropriate use can result in disciplinary actions. These and other types of online investigations also involve looking at third-party web sites and applications for actions that can be attributed to a user of interest. This paper and the associated presentation will describe how all of these types of investigations can be mislead by an incredibly common web vulnerability known as Cross Site Request Forgery. This type of forgery can be used to force a user to submit data to online web applications and also can manipulate a user's local cache and history. Using this vulnerability a user can be forced to make Internet searches, fetch arbitrary image files or web pages, post messages to online forums, or even manipulate the user's account in common web sites. Also covered will be methods to detect or rule out the use of this vulnerability during an investigation.

What Is a Cross Site Request Forgery?

A Cross Site Request Forgery (CSRF) is a type of web application vulnerability where a user can be forced to submit information to a web site of the attacker's choosing. CSRFs exploit the HTTP protocol's feature that a web page can include HTML elements that will cause the browser to make requests to other web sites. Like all HTTP transactions, the requests to the other sites will include the user's session information such as cookies or HTTP integrated authentication if they have an established session. Regardless of if the user has a session with the other sites, elements of those sites will be loaded in the victim's browser and can appear in the browser's cache and history. A CSRF can occur on an HTTP request using either the GET or the POST method.

Example Cross Site Request Forgery



Cross Site Request Forgery Hosts

An attacker does not need to lure the victim to his or her own web server to create a CSRF. Many other sites can be used to host a CSRF including online forums (which often allow a user to link to an image as an avatar or as an attachment), HTML Email, photo galleries, wikis, blogs, online auctions, and E-Commerce sites. In short, any site that allows for posting links or anything like HTML can be used as a CSRF Host. Also, a CSRF could be hosted on the target web application itself, which will usually ensure that the victim has an established session with the target application.

Cross Site Request Forgeries on an Intranet

One of the troubling aspects of CSRF is that the attacker does not even need to be able to access the application that is targeted by the forced request. All traffic to the application comes from the victim user, so as long as the victim can access the application, the CSRF can be performed. In addition, Intranet applications are often particularly vulnerable to CSRF since they often use Windows Integrated authentication so no login required and they often have poor access controls and logging.

Cross Site Request Forgeries During Investigations

During investigations and forensics, CSRFs are of interest for two reasons. First, CSRFs can cause server side state changes (which is the normal motivation for preventing CSRF in web application security). This means that actions that appear to originate from a user being investigated may have in fact been forced by another individual using a CSRF. For example, a CSRF can force a user to make search engine queries, make queries to other online sites such as photo sharing or e-commerce sites, make postings to an online forum, or make many other data submissions to sites.

Also, CSRFs are of interest during investigations because of the effects that they can have on the client web browser and the client's web traffic. A CSRF can cause sites to be visited without the user's knowledge, can cause items to be written into the user's web cache, and can cause URLs to be added to the browser history (requests forced by a CSRF may or may not appear in the browser history depends on circumstances and browser).

How Common Are Cross Site Request Forgery Vulnerabilities?

- "In fact, if you have not taken specific steps to mitigate the risk of CSRF attacks, your applications are most likely vulnerable." – Chris Shiflett in 2004: http://shiflett.org/articles/cross-site-request-forgeries

- "No statistics, but the general consensus is just about every piece of sensitive website functionality is vulnerable [to Cross Site Request Forgery]." – Jeremiah Grossman and T.C. Niedzialkowski in 2006: http://www.whitehatsec.com/home/resources/presentations/files/javascript_malware.pdf

- "Cross-Site Request Forgery (aka CSRF or XSRF) is a dangerous vulnerability present in just about every website." – Jeremiah Grossman in 2006: http://jeremiahgrossman.blogspot.com/2006/09/csrf-sleeping-giant.html

- "Cross site request forgery is not a new attack, but is simple and devastating…."
  "This vulnerability is extremely widespread…."
  "All web application frameworks are vulnerable to CSRF."
  – OWASP Top Ten 2007 at http://www.owasp.org/index.php/Top_10_2007-A5

Detecting or Ruling Out CSRF in an Investigation

In order to detect or rule out Cross Site Request Forgeries during an investigation, there are several items that can be examined. These are detailed in this section.

Look for pages that forced the requests in the web browser's cache:

- Page(s) that caused a CSRF will not always be in the cache (they could have been marked "no-cache" in the HTTP headers or in <meta equiv> tags).
- In some cases, the attacker may be using a CSRF host such as a web forum, wiki, or e-commerce application that does not allow the attacker to control caching of the page(s). In these cases, it is more likely that the page that forced the requests will be in the cache.
- Be aware of encodings of the target URL and the enclosing tag:
  - URI and enclosing tags may be encoded in a variety of manners similar to a Cross Site Scripting attack – see RSnake's Cross Site Scripting (XSS) Cheat Sheet at http://ha.ckers.org/xss.html for many examples of how the tags could be obfuscated.
  - Hostname in URL may be replaced by an IP address in one of several formats (see IP Obfuscation Calculator in the XSS Cheat Sheet for examples of these formats).

Look at the web browser's history:

- URLs that have been forced by a CSRF (such as in an IFRAME or an IMG tag) will normally not appear in the browser history (though they may in some cases, depending on the circumstances and browser).
- Pages found on disk but not in the history could be an indication of CSRF, but are more likely the result either:
  - Browser history and cache aging differently
  - User clearing the history

Construct a timeline:

- The most fruitful way to detect or rule out a Cross Site Request Forgery is to construct a timeline of the user's activity.
- Merge data from the web browser cache, web browser history, and any other logs that you may have (Proxy, IDS, Firewall, Web Server, etc).
- Examine items immediately before the activity in question to determine if a CSRF may be involved.

Look at the list of URLs that were typed into the address bar of the browser:

- This information cannot be forced.
- Not all browsers record a list of URLs that were typed into the browser.
- Users will type in the URL for only a small percentage of sites that they visit.

Look at items in browser Favorites / Bookmarks:

- This information cannot be forced.
- Users will bookmark only a small percentage of sites that they visit.

Look for evidence outside of the web browser cache:

- For example, if you are interested in image files, look for image files outside of the cache (indicating that the user intentionally saved them).
- This will only find things that the user obtained from non-web sources or that the user saved from a web site.

Determine if the web page is vulnerable to CSRF:

- This is useful if you are investigating a "traditional" CSRF issue where a state change on the server may have been forced.
- There is no way that a web application can prevent CSRFs that only aim to affect the local browser cache and history.
- If the relevant web page of application is not vulnerable to CSRF, then the information in question could not have been forced by a CSRF.
- Section below details how to determine if a page on a web application is vulnerable to CSRF.

Checking a Site for CSRF

The key characteristics of a CSRF vulnerability are that the application accepts a request that makes something occur on the server and that the attacker can determine all the parameters of that request for another user (typically in a CSRF the parameters are fixed for all users).

In order to determine if a site is vulnerable to CSRF, look for prevention mechanisms in form submissions. Form submissions that cause a state change on the server should contain an unguessable parameter contained in a hidden field or in the "action" (location where the form submits). This parameter must be something that the attacker cannot determine so he or she cannot construct a link or script to execute a CSRF.

See the Additional References section below for more information on detecting and preventing Cross Site Request Forgeries.

Additional References

For more information about Cross Site Request Forgeries, see:

- Wikipedia's entry on Cross Site Request Forgery available at http://en.wikipedia.org/wiki/Cross-site_request_forgery
- The Cross-Site Request Forgery (CSRF/XSRF) FAQ by Robert Auger available at http://www.cgisecurity.com/articles/csrf-faq.shtml
- OWASP's page on Cross Site Request Forgery (with links to information on testing for CSRF and on CSRF prevention mechanisms) available at http://www.owasp.org/index.php/Cross-Site_Request_Forgery
- Cross Site Request Forgeries in the OWASP Top Ten 2007 available at http://www.owasp.org/index.php/Top_10_2007-A5