[Bypassing JavaScript Filters – the Flash! Attack]

last updated: 05.June.2002

Obscure [obscure@eyeonsecurity.net]

EyeonSecurity
http://eyeonsecurity.net/

[Introduction to the Flash! attack]

In this document we will be describing a loophole, with security implications, found in many websites that allow Flash documents to be inserted within HTML, or uploaded to the server. This paper relies on the fact that a huge number of web surfers have installed Macromedia Flash plugin/ActiveX control, for an attacker to launch a Cross-site scripting attack. We will not go into a lot of detail in describing Cross-site scripting attacks in general; However we hope that this paper will explain how Flash documents can be used to inject JavaScript into otherwise well filtered Web Applications.

[Current web applications and cross-site scripting]

Web Applications consist of non-static web sites, which allow users to interact with the site itself[1]. Examples of such sites include Hotmail, Yahoo, MSN communities and a long list of other sites. Most of the times, this interaction involves users being authenticated, to provide a multi-user environment.

In an online community such as deviantART[2], each member has his own section and web space, where he or she can upload artistic material, such as poetry, graphics (usually jpg format), photography, and of course, Flash movies. Logged on users (as well as anonymous ones) can also view other people's work. This means that files and content are shared between different users. From a security point of view, this means that the shared content has to be trusted by both the owner of the content, and the person viewing the file.

Cross-site scripting, from now on referred to as XSS[3], is a typical attack that exploits the trust between the owner of the content and the viewer. In simple terms, XSS consists of a viewer who comes across content (set up by another possibly malicious user), which contains code, such as JavaScript, that manipulates the page to steal his session authentication, or personal details.

[Preventing Cross-site scripting attacks – current methods]

Most security aware Web Applications usually take either of the three approaches to disallow XSS attacks:

- Disallow all html code by escaping the user input.
- Allow only specific tags. This is usually achieved by making use of special code to represent specific HTML tags.
- Filter out or remove Active Scripting from HTML code.

These methods are usually believed to disallow malicious users from injecting custom HTML or Active Scripting. Web applications such as Hotmail and Yahoo Mail try to eradicate all possibilities of injection of JavaScript code (and Active content) by making use of extensive content filters. Various "Internet Authorities" and respected parties such

---

[1] What is a web application? http://davenet.userland.com/2000/03/12/whatIsAWebApplication
[2] deviantART, the largest online artistic community: http://deviantart.com/
[3] XSS: Cross Site Scripting. More information: http://www.cgisecurity.com/articles/xss-faq.shtml

as CERT[4] and Microsoft[5] have described filtering methods and the dangers of this attack in detail.

Some Web-Applications that allow Flash content to be uploaded specifically allow flash such as deviantART, others just allow files to be stored on the server for later retrieval, similar to FTP sites.

This document will describe how such content filtering can be easily bypassed because of lack of foresight in the Web Application design. The loophole described here consists of trusting Flash documents (or movies as referred by Macromedia), and therefore not treating this material as Active Content.

[The Flash! attack]

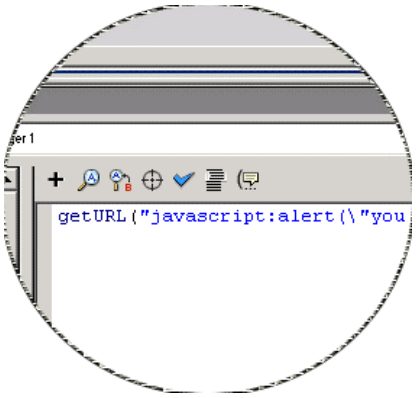
Macromedia Flash has its own built in scripting language. ActionScript[6] (the scripting language) seems very simple to seasoned JavaScript coders as it uses a very similar syntax to JavaScript, C and PERL. However this same language can be used for complex animations, simulations, creation of games etc.. What's interesting for us is the getURL() action[7]. This function allows us to redirect the end user to another page. The parameter would usually be a URL; something like "http://eyeonsecurity.net", so that the script looks like this:

```
getURL("http://eyeonsecurity.net")
```

Suppose we specify a JavaScript: URL instead:

```
getURL("javascript:alert(document.cookie)")
```

The above example pops up a JavaScript alert box with the cookie belonging to the domain hosting the page that displays the flash document. This means that we have successfully injected JavaScript by making use of "features" within Internet Explorer and Flash. In the example Flash file we insert script similar to the above in the first frame as shown in the screenshot.



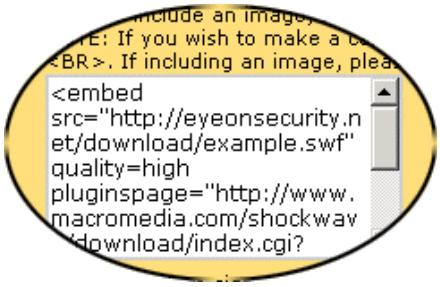
[4] CERT on XSS: http://www.cert.org/tech_tips/malicious_code_mitigation.html/
[5] Microsoft on XSS: http://www.microsoft.com/technet/security/topics/crssite.asp
[6] More information for this http://www.macromedia.com/desdev/mx/flash/
[7] documentation http://www.macromedia.com/support/flash/action_scripts/actions/geturl.html

[Example sites and software vulnerable to the Flash! Attack]

Ezboard (http://www.ezboard.com/) is probably one of the best well-known free online Bulletin Board Systems around. This BBS which is HTTP-based, allows its users to have their signatures in flash by making use of the EMBED tag. Therefore in our tests we edit our preferences and specify the following code in the signature:

```
<embed
        src="http://eyeonsecurity.net/download/example.swf"
        pluginspage="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=ShockwaveFlash"
        type="application/x-shockwave-flash"
        width="0"
        height="0"
>
        </embed>
```

The below screenshot illustrates the idea better.



This code will be added to each post the attacker submits on the Ezboard forum, allowing him to steal the user's session cookie.

DeviantART which is a very popular website, encourages it's users to submit flash animations and creations to be viewed by other site members. Of course a malicious user with intent to steal user accounts and possibly administrative accounts, would create a new account, upload a malicious Flash file and wait for the results. No demonstration is available for this site.

MSN Communities[8] – this site allows users to upload their own files … amongst the files we uploaded were SWF files, which in turn execute JavaScript code. This is a very obvious security flaw. In a previous paper[9] on EyeonSecurity, named "Microsoft Passport Account Hijack Attack", we outline how a single flaw in an MSN or Passport network site creates a significant security problem.

Anonymous services such as Anonymizer[10] and The-Cloak[11], are also vulnerable to this attack. These services try to filter out JavaScript from HTML pages, however fail to recognize the attack described here at the time of writing. Meaning that web master

[8] MSN Communities: http://communities.msn.com/
[9] EyeonSecurity Papers: http://eyeonsecurity.net/papers/
[10] Anonymizer is a commercial service: http://anonymizer.com
[11] The-cloak is a free service for hiding your ip and more: http://www.the-cloak.com/

linking (or redirecting) its users to an SWF file can bypass the restrictions set up by these services.

Two specific Forum (BBS) software, which are particularly vulnerable to this attack, are Ikonboard and YaBB[12]. These particular forums allow only specific custom tags which are then parsed by the Web Application to produce the end result. However these forums allow flash animations to be embedded within the page by using the [flash] special tag, which is converted to the correct Object tag.

Example

```
[flash]http://eyeonsecurity.net/download/example.swf[/flash]
```

The above would be interpreted by the script and transformed to:

```
<object
        classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
        width=200
        height=200>
<param
        name=movie
        value=http://eyeonsecurity.net/download/example.swf>
<param
        name=play
        value=true>
<param
        name=loop
        value=true>
<param name=quality
        value=high>
<embed
        src=http://eyeonsecurity.net/download/example.swf
        width=200
        height=200
        play=true
        loop=true
        quality=high>
</embed>
</object>
```

Of course these specific examples are not the only vulnerable systems around. Any online service, which allows Flash content to be inserted is vulnerable to XSS attacks. The vendors and services described in this section have been notified of the flaw before this document has been made public. This means that the specific examples outlined in this section might have been fixed when you are reading this.

[12] Ikonboard: http://www.ikonboard.com/ ; YaBB: http://www.yabbforums.com/

[Fixing the issue]

Simple solution: DO NOT ALLOW FLASH FILES IN YOUR WEB APP.

However in most cases, the solution is not that simple. Consider the case for deviantART for example. Flash animations are part of the content of the site. Such content is considered critical for the Flash community within deviantART.

Possible solutions for:

Macromedia (Flash player developer)

Macromedia and EyeonSecurity have worked together to provide a solution for Web developers. It was suggested to allow Web designers to change the behavior of embedded Flash content within HTML pages. This solution addresses issues within forums and similar sites, but is designed not to break any exist animations/flash movies.

However such a solution does not address websites such as MSN Communities and deviantART. These sites allow users to upload SWF files rather than just link to them. Macromedia (as well as EoS) actively discourages web application design that allows users to upload unchecked Flash content.

It must be noted that interaction between the HTML page (and JavaScript or other Active Content for that matter) and a Flash file is supported using different functions[13], and the method described in this document is a "hack" rather than a supported function. However a well-known application that produces Flash Movies, called Swish[14] makes use of the JavaScript method to allow Web Designers insert their own JavaScript code.

Web Developers and Web Designers

A good solution would be to actually parse the flash animation and filter malicious parameters in getURL(). This addresses the case when a Web application allows SWF files to be uploaded to the server. Webmasters are highly encouraged to parse and filter Flash content if they allow users to upload. Webmasters may choose to block any Flash content which contains getURL() actions that do not specifically point towards an HTTP site. Another solution would be to change all getURL() actions to point to a new window. This can be achieved by specifying the target window as "_blank". By making the described changes, JavaScript URLs will not execute under the hosting domain's privileges.

Yet another possibly more feasible solution (for certain webmasters) would be to make use of a separate domain for storing and displaying the Flash movies. This method may also be used for other documents, such as HTML files, to allow Active Content without enabling attackers to obtain the session authentication by stealing the cookie. This means that if your domain is

[13] An example of communication between JavaScript and Flash: http://www.macromedia.com/support/flash/ts/documents/java_script_comm.htm

[14] Swish is an alternative to Macromedia's Flash MX http://www.swishzone.com/

“securewebapplication.com”, you could store possibly malicious content on “securewebapplication.net”. Of course this means that the content of “securewebapplication.net” does not require session authentication and therefore this content is shared with anonymous users. It is important to note that the malicious content should only be displayed from a “sanitized” domain, meaning that if the flash document is contained in an HTML file, it has to be also displayed from the “sanitized” domain.

Suggestions for users of specific products/services

Ezboard.
The response given by the product support of Ezboard was the following:

```
Unfortunately, it is very, very tough to make it impossible to make
cookie stealing impossible.  If we did, ezboard would not have nearly as
many customization options as it currently does.  Fortunately, we have a
high security login setting which checks the user's IP address.  If it is
selected, nobody else will be able to use the cookie to login while the
real user is logged in.
```

This may a good solution, which addresses XSS issues, including the one described in this paper. However IP checking does not work well with Internet users behind proxies.

YaBB
The solution response by Corey Chapman was:

```
We didn't build it as an option, but we've informed people to simply
comment out or delete the (I think 2) flash-rendering lines in
yabbc.pl.  I'll probably build a disable feature in for the next
update, assuming there'll be one for this version.
```

This is quite straightforward solution. Of course we suggest also removing the Flash icon display when editing the message.

Ikonboard
This product allows administrators to disable flash support, according to Andrew (Ikonboard).

```
This does open the
security hole that you mentioned, as people have used it before to
change people's avatars. The problem is that we can't just remove the
option as many users would be very unhappy with us. As it is right now
you can disable this option, and not allow flash on your board.
```

This sounds like a neat solution – however we did not test this feature.

[Demo! on EyeonSecurity.net]

For a demonstration of the issue described here check out:

http://eyeonsecurity.net/advisories/flash-demo/

[References]

Cross-site scripting

XSS Faq by CgiSecurity.com
http://www.cgisecurity.com/articles/xss-faq.shtml

Information on Cross-Site Scripting Security Vulnerability by Microsoft
http://www.microsoft.com/technet/security/topics/crssite.asp

CERT
Understanding Malicious Content Mitigation for Web Developers
http://www.cert.org/tech_tips/malicious_code_mitigation.html

Malicious HTML Tags Embedded in Client Web Requests
http://www.cert.org/advisories/CA-2000-02.html

Evolution of Cross-Site Scripting Attacks by iDefense
http://www.idefense.com/XSS.html

Web Application Security – General information

Open Web Application Security Project
http://www.owasp.org/

Web Application Security - PowerPoint Presentation
http://www.whitehatsec.com/dc9.html

Flash Documentation

Designer & Developer Center
http://www.macromedia.com/desdev/mx/flash

Flash Kit – A Flash Developer Resource
http://www.flashkit.com/

OpenSWF.org – information about the Flash format
http://www.openswf.org