

Forum:[Hack Forums](#)

Thread: [\[TUTORIAL\] SWF / Flash-based Exploitaiton - THE BASICS \[PART 1\]](#)

M L T - December 10, 2018, 3:42 am

Flash Exploitation / Cross-Site-Flashing (XSF) Tutorial:

Introduction:

To fully understand this tutorial, you'll need basic knowledge of XSS and how SWF's work in general. It is mainly going to be covering vulnerabilities that can result in javascript being executed on the target site via an insecure SWF. This tutorial will cover the bare basics.

When testing in the wild, I think these are a very good thing to look into. These vulnerabilities are generally pretty easy to find, and many big sites appear to be vulnerable. Although flash is a dying technology, you should take into account the fact that it is still enabled by default within Google Chrome therefore a lot of people can potentially be affected by flash vulns.

Within a week of testing via these methods, I identified vulnerabilities in Microsoft, Bank of America, Dell, HP, Oracle, Intel, Forex, NASDAQ, JPMorgan Chase, NASA, Samsung, Sony, Kodak, Adobe, CNN, Washington Post, US DoD/Army/Navy/Marines/Airforce, United Nations, whitehouse.gov, norad.mil, java.net and more.

For those who have worked with ActionScript before, you're probably aware of how similar it is to javascript (both being ECMA-based) in terms of syntax. Like other languages that can be vulnerable to XSS, failure to properly sanatize user inputs within actionscript can allow these vulnerabilities to take place – in addition to this there are a few vulnerabilities specific to actionscript, such as cross site flashing and `externalInterface.call()`; fuckery.

List of unsafe methods in ActionScript:

Quote:
`loadVariables()`
`navigateToURL()`
`getURL()`
`getURLBlankVar()`
`getURLParentVar()`
`getURLJSParam()`
`loadMovie()`
`loadMovieVar()`
`loadMovieNum()`
`FSScrollPane.loadScrollContent()`
`LoadVars.load`
`LoadVars.send`
`XML.load ( 'url' )`
`XML.sendAndLoad ( 'url' )`
`LoadVars.load ( 'url' )`
`LoadVars.send ( 'url' )`
`Sound.loadSound( 'url' , isStreaming );`
`NetStream.play( 'url' );`

```
flash.external.ExternalInterface.call(_root.callback)
externalInterface.addCallback
htmlText
htmlVar
loadClip
AddDLL
```

Crossdomain.xml: Crossdomain policies are a set of rules that determine which clients (such as flash player) can access data across multiple domains. For a full explanation of how crossdomain policies work, you can read the specification [here](#)

To exploit the methods I'm going to describe, you'll need your crossdomain policy (hosted on your site and named crossdomain.xml to look something like this:

```
Quote:<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.adobe.com/xml/dtds/cross-domain-policy.dtd";
<cross-domain-policy>
<site-control permitted-cross-domain-policies="all"/>
<allow-access-from domain="*" secure="false"/>
<allow-http-request-headers-from domain="*" headers="*" secure="false"/>
</cross-domain-policy>
```

I'll explain why this is useful for exploitation soon, but first i'm going to cover some more traditional XSS that flash can be susceptible to.

Whitebox approach to SWF analysis: SWF files can be easily decompiled, meaning that with some basic actionscript knowledge it is always possible to perform a whitebox code audit on the SWF. There are many free tools available for SWF decompilation, but if you're a Linux user then [Flare](#) would be my recommendation due to its simplicity and the fact that it is far more lightweight than most other flash decompilers.

To save the raw SWF there are a few options, you can either get the direct link and wget it to your box (if its a HTML page then you'll need to view the page source and CTRL + F for the SWF, it will be stored within an <object> tag – it is also worth doing this anyway even with a blackbox approach so you can see any associated flashvars and what the status of allowScriptAccess is set to) or you can use your browser and select the 'save page as' (CTRL+S) option which will save the SWF for you.

Once decompiled, it will be generally saved in .fla format (or .flr if you're using flare), you can then open it with a text editor and perform analysis of the source code in order to hunt for potential vulnerabilities (by checking for functions that could be vulnerable + checking for undefined flashvars and just getting a general understanding of the code flow)

Flashvars + Undefined variables: Flashvars are variables that can take user-supplied input for the flash application for use within the application. Generally they are initialized within HTML via embed/object like so:

```
Quote:<embed src="lol.swf" width="1337" height="1337"
FlashVars="param=something&param2=somethingelse&param3=lol">
```

Within AS3 a developer needs to assign their flashvars to local variables, but within AS2, any global variable that is not initialized is automatically assumed to be a flashvar and treated as such. This means that if you find a global variable that has not been explicitly defined within the source that you're auditing, then you can assign your own values to that variable by calling it via GET like you would with a flashvar. Depending on the context in which this variable is meant used within the program, this may or may not lead to a multitude of vulnerabilities.

You can check for global variables by looking for strings preceded with any of the following:

Quote: _root
_global
_level0

If any of those are declared but not defined, then there is a good chance of a potential vulnerability being present.

getURL() / NavigateToURL(): If getURL or its AS3 equivalent is setup in an incorrect manner, then exploiting it is very trivial. It does as the function name suggests (attempts to get a movie from a remote URL) so you can just exploit this to result in XSS in the same way as you would with an open redirect by calling javascript:alert(1337) or data:text/html, <script>alert(1337)</script> – also remember that if HTML tags trigger WAF** then you can base64 encode the input like so:
data:text/html;base64,PHNjcmlwdD5hbGVydCgxMzMKTWvc2NyaXB0Pg==

If you decompile an SWF and see a flashvar which you can control being used as the argument for getURL, then you can probably exploit it by calling javascript:// or data://

externalInterface.call: This function can be abused for callback functionality – for those who may not be familiar with exploiting callback functionality in actionscript, let me give an example in another language that people may be more familiar with.

Within PHP, PRCE functions can be abused for callback functionality, allowing you to call phpinfo(); to disclose the uname output, php version, etc etc. The most common example of this would be preg_replace(); with the /e modifier active. Actionscript can be vulnerable to something similar (except you're obviously not calling phpinfo();) – externalInterface.call takes two arguments – the name of the JS function to call, and the string to pass to that function. In the case of externalInterface.call, callback functionality can be abused in order to trigger an XSS.

a typical vector used for exploiting externalInterface.call for XSS would look something like this:

Quote: \");throw_error()}catch(e){alert(document.domain)}}//
\\");function%20someFunction(a){}prompt(1)//

although in some cases, a simple alert`0` (with template strings) or \");alert(document.domain); will suffice.

HTML injection / Callback functionality via textfield objects: Textfield objects within flash allow HTML to be rendered but in a very limited sense, it will allow certain tags to be rendered. To my knowledge the only tags that can be used to trigger XSS via this are <a> and – if the <a> tag does not allow you to set hyperlink to a javascript: uri then you can use data: with a base64 encoded input.

In older versions of flash player you can trigger XSS via the src attrib for by appending an SWF extension to your payload:

Quote:

it is also possible to set the src attrib of the image to a malicious SWF designed to trigger a js alert.

You can also use the <a> tag to exploit callback functionality by calling both swf public functions and native static asfunctions like so (to circumvent crossdomain policies in this example):

Quote:

asfunction can be used to achieve XSS also, by calling a native function such as getURL like so:

Quote:

Injecting HTML/JS Via CDATA: CDATA stands for character data and it allows you to include data that could be considered as XML markup but should not be. For example lets say you're crafting a malicious XML file to trigger an XSS in a vulnerable SWF and you see a <text> section where you want to inject some HTML that the victom can interact with in order to trigger your javascript payload. If you added your HTML content in its regular form between the <text> tags then it would just be treated as more XML tags (due to the flexibility of XML and the fact that any tag name can be considered potentially valid). You use CDATA so it knows that your data should not be treat as XML markup, meaning you can add your own HTML markup and it will be treated as valid HTML when rendered. Here is an example of how you would add some HTML via CDATA:

Quote:<![CDATA[<h1>My HTML content</h1>]]>

Loading Malicious XML Files: In some cases, you'll find files with an external XML config file passed as a flashvar via GET to the SWF. This config file can be used to control both the style and the content of the page. In order to do this, you will need to have a crossdomain policy setup (as described above) on your site. This is generally due to an insecure implementation of loadMovie or XML.load although it can also occur as a result of client-sided javascript being used to control the flash movie (e.g. via getVariable() / setVariable())

In order to exploit this you need your XML file to conform to their style (so its best to have a copy of the XML file they are using to base your payload off of) and you'll also need a correctly configured crossdomain policy on your site.

root@HF - December 10, 2018, 4:58 am

i reli love ur tutorials babe i never knew how easy it was to exploit flash i never tried before but with this part 1 and part 2 i feel like i learned so much more than even univeristy teaches at time. i didnt realise u could exploit flash using xss its crazy <3 keep up the good work babe <3

(Dec 10, 2018 04:42 AM)M L T Wrote: [\[CITE\]](#) <3

Temp_reassign - December 10, 2018, 5:02 am

Yea, part 2 brought me back here. good shit bro.

M L T - December 10, 2018, 5:35 am

(Dec 10, 2018 05:58 AM)Spectacular.blondhey Wrote: [\[CITE\]](#) i reli love ur tutorials babe i never knew how easy it was to exploit flash i never tried before but with this part 1 and part 2 i feel like i learned so much more than even univeristy teaches at time. i didnt realise u could exploit flash using xss its crazy <3 keep up the good work babe <3

Thanks :) in parts 3 and 4 I will begin to go wayyyy more in-depth with regards to flash exploitation. So far I've pretty much just covered the bare minimum.

root@HF - December 10, 2018, 5:37 am

(Dec 10, 2018 06:35 AM)M L T Wrote: [\[CITE\]](#) Thanks :) in parts 3 and 4 I will begin to go wayyyy more in-depth with regards to flash exploitation. So far I've pretty much just covered the bare minimum.

please message me when u release ur part 3 and 4 im so eager to see progression here <3

M L T - December 10, 2018, 5:39 am

(Dec 10, 2018 06:37 AM)Spectacular.blondy Wrote: [\[CITE\]](#) please message me when u release ur part 3 and 4 im so eager to see progression here <3

I'll be releasing em later today :) writing them up currently. I'm going to be covering deep packet inspection / packet manipulation of flash games, etc... before diving deeper into 0days for flash player itself, allowing you to infect people.

root@HF - December 10, 2018, 5:45 am

(Dec 10, 2018 06:39 AM)M L T Wrote: [\[CITE\]](#) I'll be releasing em later today :) writing them up currently. I'm going to be covering deep packet inspection / packet manipulation of flash games, etc... before diving deeper into 0days for flash player itself, allowing you to infect people.

wow this is crazy babe i love it. would this be using wireshark to inspect and manipulate the packets?

u got me hooked im eagerly waitin babe. the 0days for flash player will be very interesting babe <3