# Flash

## Challenges

**Flash - Authentication :** https://www.root-me.org/en/Challenges/Web-Client/Flash-Authentication

Use a SWF decompiler : JPEXS , Sothink , TrillixFlashdecompiler
Check the code then brute force.
This is an annoying challenge and hard (For me)

## Notes

Adobe Flash is a technology used to create video, animations and other rich content. Flash files can be used as standalone applications or be embedded in HTML pages.

Build the logic using ActionScript, very similar to JavaScript.

Flash ActionScript needs to be compiled before execution. The compiled object is in **SWF format** (Shockwave Flash).

decompilers:
• Sothink Flash decompiler : http://www.sothink.com/product/flashdecompiler/
• TrillixFlashdecompiler : http://www.flash-decompiler.com/

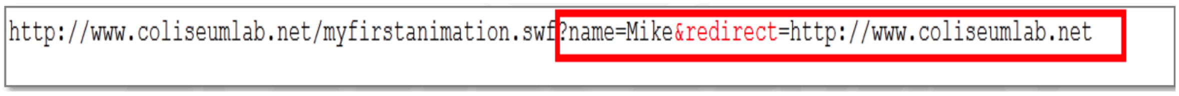
To embed a Flash animation within a web page, you can use an object element with type application/x-shockwave-Flash.

The HTML page can embed the Flash animation and specify a param element with the attribute allowScriptAccess to describe the policy for communication between the animation and the parent HTML page.

allowScriptAccess can have three values:
1- Always :
The SWF file can always communicate with the HTML page regardless of the domain(SWF can be on a domain A and communicate with domain B).
2- sameDomain :
The SWF file can communicate with the HTML page only if they share the same domain.
3- Never:
The SWF file can never communicate with the HTML page

There are three methods by which you can pass arguments to Flash files.

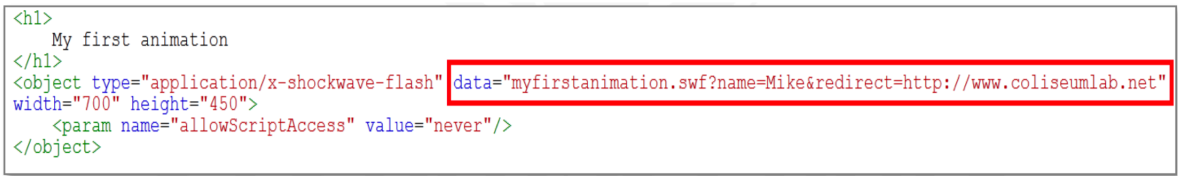
Method 1 – Direct reference :
Flash is not embedded in an HTML page and arguments are provided directly through the URI. When the browser directly loads the URI, a dummy HTML page is created.



```
http://www.coliseumlab.net/myfirstanimation.swf?name=Mike&redirect=http://www.coliseumlab.net
```

Method 2 – Flash Embedded in HTML page

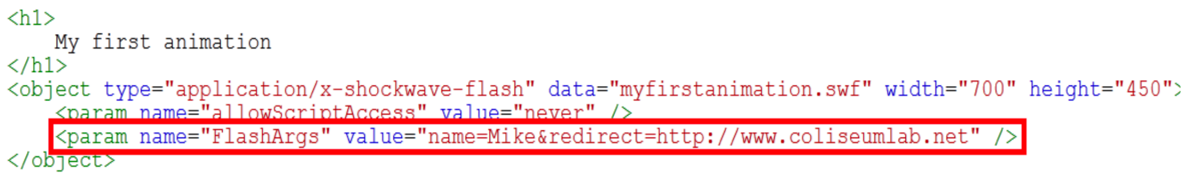
Flash is embedded in an HTML page and arguments are passed through the attribute data of the object element.



```
<h1>
    My first animation
</h1>
<object type="application/x-shockwave-flash" data="myfirstanimation.swf?name=Mike&redirect=http://www.coliseumlab.net"
width="700" height="450">
    <param name="allowScriptAccess" value="never"/>
</object>
```

Method 3 – FlashArgs attribute

Flash is embedded in the HTML page and arguments are passed through the attribute FlashArgs of the object element.



```
<h1>
    My first animation
</h1>
<object type="application/x-shockwave-flash" data="myfirstanimation.swf" width="700" height="450">
    <param name="allowScriptAccess" value="never" />
    <param name="FlashArgs" value="name=Mike&redirect=http://www.coliseumlab.net" />
</object>
```

Flash Security Model:

Flash Sandbox:
By default, SWF files hosted on the subdomain a.example.com can only access resources (SWF files, images, sounds, text files, etc.) on a.example.com.

Flash Stakeholders:
When a Flash animation is started by a player, different security controls are performed at different levels.
Stakeholders describe different roles (human or not) involved in the control of security settings.

There are four stakeholders:
administrator --> user --> website --> author.

Flash security checks are performed, starting from the first stakeholder (Administrator) and working to the last. An operation fails when the first stakeholder in the chain blocks it.
Each stakeholder can enforce security controls using different means.

**Administration role:**
represented by the system administrator
responsible for the Flash player installation
configures Flash security settings that will affect all operating system users

mms.cfg :
This text configuration file is read when Flash player starts. It allows administrators to grant or restrict access to a variety of features. In Microsoft systems, it is generally located in the directory system32\Macromed\Flash.

The Global Flash Player Trust directory:
Administrators can register SWF files as trusted for all operating system users; this means that SWF files can interact with other SWF files and can load data from local and remote locations.

**User role:**

represented by the operating system user running a Flash animation through the player.

Settings manager, Settings UI :
The Settings Manager and Settings UI provide security-related options such as: camera and microphone settings, shared object storage settings, settings related to legacy content, and so on.

User Flash Player Trust directory :
Users can register SWF files as trusted; this means that SWF files can interact with other SWF files and can load data from local and remote locations.

**Website role:**
represented by the web server component responsible for the delivery of the Flash animation and/or Flash resources to the client.

This role can decide if an external SWF animation can interact (through ActionScript) with data available on the web server.

*Policy File :*
A policy file is a file placed in a specific location on the web server.
This file is read by the Flash player; SWF files residing on third party domains will use this to derive their access rules, which they must adhere to.
This policy file can grant or deny access to resources stored on the current server enforcing the policy.
The resources that pertain to this access are SWF (images, text files, or other SWF files).

Example:
A SWF animation (on domain: A.com) tries to access external resources (images and audio files) available on a different domain (B.com).
The policy file (crossdomain.xml) on domain B.com will determine if the operation is allowed or not.

Depending on the requested resource, ActionScript can initiate two different types of connection:

- Document-based server connection for ActionScript objects
  - Loader (to load SWF files or images JPG, PNG, GIF)
  - Sound
  - URLLoader (to load text or binary data)
  - URLStream
- Socket connection for ActionScript objects
  - Actionscript socket
  - XMLSocket

URL Policy File:
This policy file is checked by the Flash player if it is a document- based server connection.
By default, when the Flash file accesses external resources like images, text, sounds, etc., Flash player looks for a file named crossdomain.xml in the root directory of the server.

A SWF file can check for policy files in different locations by calling the Security.loadPolicyFile() method.

By default, the Flash player looks for a socket policy file on port 843; we refer to this policy file as the Master policy file.

**Author Role :**
The author role is represented by the developer of the Flash  animation.
This role can affect the interaction between SWF files available on different domains.

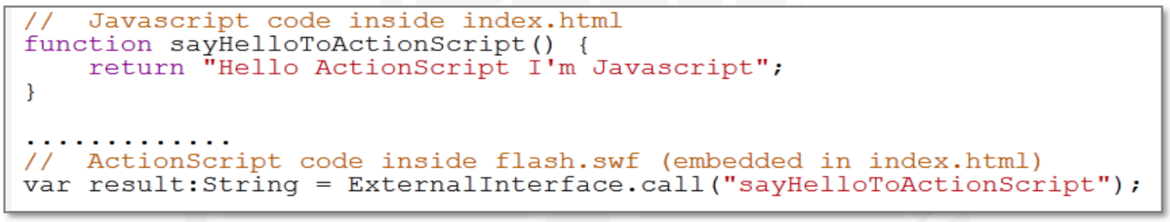
The API Security.allowDomain(<allowedDomain>) grants
permissions to the following features:
• The interaction between different SWF files (Flash Cross- Scripting)
• Display list access
• Event detection
• Full access to properties and methods of the Stage object

**Flash cross scripting**
By default, a SWF file cannot access the ActionScript properties and methods of another SWF file available on a different domain.
Access is granted only if the accessed SWF allows the domain of the calling SWF.

To achieve this goal, the API call Security.allowDomain() can be used by the accessed SWF ActionScript.

Example: An external SWF animation (domain: A.com) accesses properties and methods of another SWF file available on a different domain (B.com).

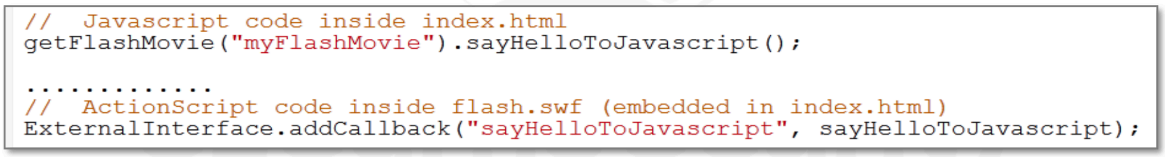
**Calling JS**
When Flash animations are embedded in HTML pages, ActionScript code (available in the SWF file) can call the parent page's JavaScript code by using the ExternalInterface class.



```
//  Javascript code inside index.html
function sayHelloToActionScript() {
    return "Hello ActionScript I'm Javascript";
}

.............
//  ActionScript code inside flash.swf (embedded in index.html)
var result:String = ExternalInterface.call("sayHelloToActionScript");
```

**Calling ActionScript:**
When Flash animations are embedded in HTML pages, JavaScript code can call Actionscript code included in the embedded Flash file.



```
//  Javascript code inside index.html
getFlashMovie("myFlashMovie").sayHelloToJavascript();

.............
//  ActionScript code inside flash.swf (embedded in index.html)
ExternalInterface.addCallback("sayHelloToJavascript", sayHelloToJavascript);
```

The method has the following definition - navigateToURL(URLRequest, [target]).
This ActionScript method is used to open a URL in the window embedding the Flash file. It is similar to the JavaScript function window.open. The function has an optional second parameter - target. This parameter can be used to specify where the URL should be opened (a given frame, current window or a new window).

The parameter URLRequest can contain JavaScript code, such as javascript:alert('HellofromActionscript.'); in this case, the JavaScript code will be run only if one of the following conditions is true:

- The SWF file is a locally trusted file
- The target is the HTML page in which the SWF file is embedded, and the allowScriptAccess rules described above apply
- The target and the SWF file are part of the same sandbox (they are located in the same subdomain)

If input sanitization does not occur, this method can be exploited by malicious users, allowing attacks on other users through Flash applications.
http://web.archive.org/web/20170328012534/https:/help.adobe.com/en_US/ActionScript/3.0_ProgrammingAS3/WS5b3ccc516d4fbf351e63e3d118a9b90204-7c9b.html

**Local Shared Object**
Flash features an internal storage mechanism based on Local Shared Objects.

Conceptually, Local Shared Objects are similar to browser cookies:
- They can be used to track user activity or to store preferences.
- They are read-only by subdomains that have set them.

But:
- They are not sent back and forth over HTTP connections.
- They do not expire.
- They are stored in a dedicated Flash directory and are shared by all browsers on a system.
    - Example: a local shared object set through the Mozilla browser by a domain example.com can be read by a Chrome browser (installed on the same machine) visiting the same domain. (Cross-browser access.)
    - Path in Windows 7: C:\Users\<user>\AppData\Roaming\Macromedia\Flash Player\#SharedObjects\
- They can contain complex (and large amounts of) data, so they offer the advanced features of local storage.

With Action Script
var myCookie:SharedObject = SharedObject.getLocal("information");

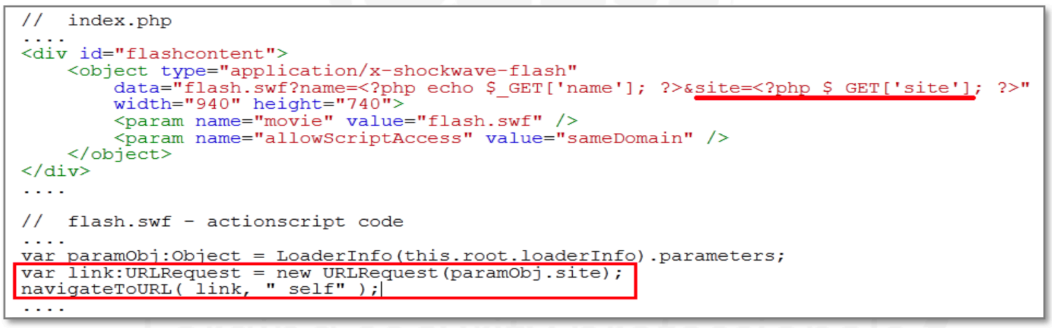
If the shared object information exists, its properties are retrieved and saved into the object myCookie; otherwise, an empty sharedObject will be created.

## Flash Attacks

### Flash Parameter Injection

When:
- The attacker can insert malicious code into the web application.
- The web application passes the input provided by the attacker to the Flash animation without any significant sanitization.
- The Flash animation, embedded in the HTML page, does not sanitize input parameters.
- The SWF source code allows HTML injections or XSS.

Example:



```
//  index.php
....
<div id="flashcontent">
    <object type="application/x-shockwave-flash"
        data="flash.swf?name=<?php echo $_GET['name']; ?>&site=<?php $_GET['site']; ?>"
        width="940" height="740">
        <param name="movie" value="flash.swf" />
        <param name="allowScriptAccess" value="sameDomain" />
    </object>
</div>
....

//  flash.swf - actionscript code
....
var paramObj:Object = LoaderInfo(this.root.loaderInfo).parameters;
var link:URLRequest = new URLRequest(paramObj.site);
navigateToURL( link, " self" );
....
```

Index.php :
This is the container page. This file processes an input argument named site (taking it from the URL of the received request).
User input, in this case, is passed to the embedded Flash without sanitization.

Flash.swf:
This file contains ActionScript code which processes the input argument without any sanitization.
Flash code uses the site input argument to replace the current window.
But, if the site argument contains JavaScript code, it will be run as if it was executed from the container page (index.php).

If you want to read cookies set by the current domain (where
index.php is located), you should use the following payload:  (simple XSS attack run through Flash. )
**site=javascript:alert(document.cookie);**

**Fuzzing Flash**

SWFinvestigator is an interesting tool which allows researchers to analyze the quality and the security aspects of SWF files.
It contains a fuzzer that searches for XSS vulnerabilities.

Features:
- Actionscript2/3disassembling
- SWFtagviewing
- Local Shared Object (LSO) analyzing
- Dynamic function calling

**Hardcoded Sensitive Information**

As we said earlier, Flash contents need to be compiled before they can be run.

Many Flash developers believe that it is impossible to obtain the source code of the Flash application once compiled. They are often misled by this idea, and may, in turn, hardcode information:
• URLs of hidden resources
• Resources that should be hidden to regular users
• Credentials of services

An attacker could use a decompiler to decompile the SWF files, obtain the source code and find credentials and other useful information.

## Flash Pentest Methodology:

Main areas to check are:
1. Client-side components (SWF files and container page)
    - Static analysis of the SWF source code:
        - Obtain the source code of the SWF files and search for interesting hard-coded information (URL, credentials info, etc.).
        - Check if input parameters are sanitized.
    - Analysis of the container page (generally the HTML page containing the Flash SWF file):
        - ChecktheallowScriptAccessparameter.
        - Check if input arguments that will be passed to the Flash are sanitized.

    - Analysis of the website hosting the Flash application
        - Check if the policy file (crossdomain.xml) is configured properly.
    - Search for common vulnerabilities
        - If the input is not sanitized, you must check whether an attacker can take advantage of it. Check for common vulnerabilities (HTML injections, XSS). You could use the SWFinvestigator’s XSS fuzzer.

2. The Communication protocol between the client side player and the server-side components
    - each request could be sent according to a given protocol:
        - SOAP
        - AMF
3. Server-side components
    - In order to enumerate all backend functions (services), we have to test each of them against common server-side vulnerabilities (SQL injections, RFI, etc.).