Flash it baby!

Finding vulnerabilities in SWF files (v2.0)

Soroush Dalili @irsdl
v2.0

August 2016
@BSidesMCR
Just a Rookie Track!

whoami

You need adblock++!

- Security consultant at NCC Group
- +10 years in web application security
- Researcher and bug hunter (I am trying to be?!)

- @irsdl
- https://soroush.secproject.com/blog/

Not Yet!

Flash Isn't Quite Dead Yet!

- They ignore it, they laugh at it, but they have to fight it!
- They may not use it, but probably have it!
- SWF in JS libraries, HTML WYSIWYG editors, Players in CMSes, …
- XSS is XSS no matter where it is!

What's on the Menu Today?

- Assumptions:
  - Client-side web application issues
  - SWF files in browsers via a website (not local with file system nor AIR apps)

- Excluded:
  - Making a website vulnerable by uploading a Flash file
  - Exploiting a website by creating a reflected Flash file (e.g. Rosetta Flash)
  - Attacking server-side
  - Nudity!!!

Where is the naked photo? Bo0o0o!

Introduction

Easy stuff yo!

- ActionScript is based on ECMAScript 😍
- .SWF -> A compiled Flash file (binary) -> We care about this ❤
- Versions: 1 and 2 ;then 3 to supports object oriented designs 💞

Embedding into a HTML Page

- Embedded via OBJECT or EMBED tags
  - Example with OBJECT:

```
<object type="application/x-shockwave-flash" data="file.swf">
        <param name="movie" value="file.swf" />
        <param name="FlashVars" value="param1=value1&p2=v2" />
        <param name="allowscriptaccess" value="always" />
</object>
```

  - Example with EMBED:

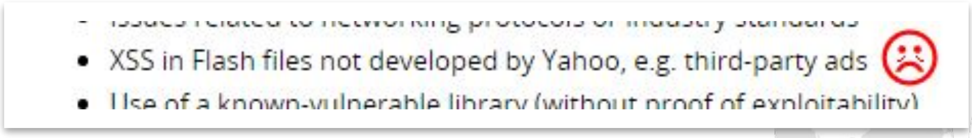
```
<embed src="file.swf" type="application/x-shockwave-flash"
allowScriptAccess="always" FlashVars="param1=value1&p2=v2">
```

- “OBJECT” can accept “allowScriptAccess” as attribute -> Not IE
- Use “TYPE” when content-type is different
- “classid”, “codetype” -> obsoleted since HTML5
- “allowScriptAccess=always” to communicate with HTML!
- “allowScriptAccess=samedomain” is default!

Bug Hunting Strategy

- Finding Flash Files
  - Google... filetype:swf site:example.com
  - Download open source apps/libs
  - Search in contents for SWF
  - Search similar open source projects for SWF
- Search for known issues
- Automated testing
- Manual testing

- Note: Is it eligible in bug bounty?
  - e.g.: https://hackerone.com/yahoo



- Issues related to networking protocols or industry standards
- XSS in Flash files not developed by Yahoo, e.g. third-party ads ☹
- Use of a known-vulnerable library (without proof of exploitability)

What Type of Issues?

- Insecure crossdomain.xml
- CVE-2011-2461 – still Alive!
- Vulnerabilities in SWF Files
  - Cross-site scripting (XSS)
  - Cross-site data hijacking (XSDH?)
  - Same Origin Method Execution (SOME)
  - Open redirections (doesn't have a fancy name!)
  - Information disclosure - credentials, hidden URLs, etc.
  - Spoofing/Defacement via loading remote objects
  - Storing sensitive data on the client-side
  - Log forging (not really important most of the times)

Insecure crossdomain.xml

- Least restrictive policy:

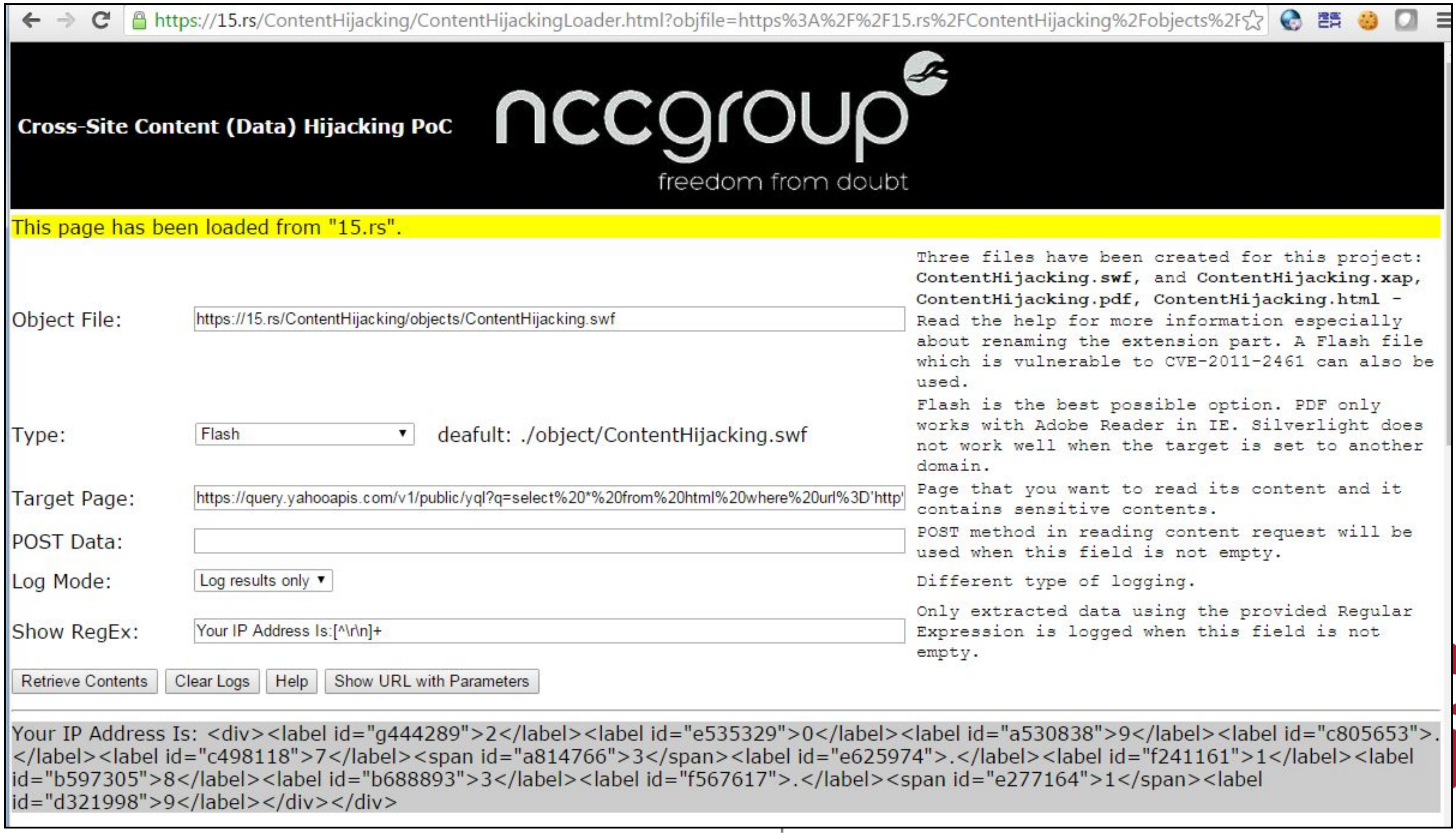
```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.adobe.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
    <site-control permitted-cross-domain-policies="all"/>
    <allow-access-from domain="*" secure="false"/>
    <allow-http-request-headers-from domain="*" headers="*"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

- “crossdomain.xml” instead of “clientaccesspolicy.xml” for Silverlight:
  - The most secure one is insecure!

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
   <allow-access-from domain="*" headers="SOAPAction" secure="true">
</cross-domain-policy>
```

Content Hijacking PoC Tool

- Cross-Site Content Hijacking (XSCH) PoC:
  - https://github.com/nccgroup/CrossSiteContentHijacking
  - E.g.: *https://query.yahooapis.com/crossdomain.xml*



https://15.rs/ContentHijacking/ContentHijackingLoader.html?objfile=https%3A%2F%2F15.rs%2FContentHijacking%2Fobjects%2F

Cross-Site Content (Data) Hijacking PoC

nccgroup freedom from doubt

This page has been loaded from "15.rs".

Three files have been created for this project: **ContentHijacking.swf**, and **ContentHijacking.xap**, **ContentHijacking.pdf**, **ContentHijacking.html** - Read the help for more information especially about renaming the extension part. A Flash file which is vulnerable to CVE-2011-2461 can also be used.

Object File:	https://15.rs/ContentHijacking/objects/ContentHijacking.swf	
Type:	Flash — deafult: ./object/ContentHijacking.swf	Flash is the best possible option. PDF only works with Adobe Reader in IE. Silverlight does not work well when the target is set to another domain.
Target Page:	https://query.yahooapis.com/v1/public/yql?q=select%20*%20from%20html%20where%20url%3D'http'	Page that you want to read its content and it contains sensitive contents.
POST Data:		POST method in reading content request will be used when this field is not empty.
Log Mode:	Log results only	Different type of logging.
Show RegEx:	Your IP Address Is:[^\r\n]+	Only extracted data using the provided Regular Expression is logged when this field is not empty.

```
Your IP Address Is: <div><label id="g444289">2</label><label id="e535329">0</label><label id="a530838">9</label><label id="c805653">.</label><label id="c498118">7</label><span id="a814766">3</span><label id="e625974">.</label><label id="f241161">1</label><label id="b597305">8</label><label id="b688893">3</label><label id="f567617">.</label><span id="e277164">1</span><label id="d321998">9</label></div></div>
```

CVE-2011-2461 - The Dead is Alive!

- Flex SDK issue (between 3.x and 4.5.1)
- A new input to load external SWF files
- Attacks:
  - Same-Origin Request Forgery
  - Cross-Site Content Hijacking

- Found by Mauro Gentile (@sneak_) & Luca Carettoni

Finding CVE-2011-2461

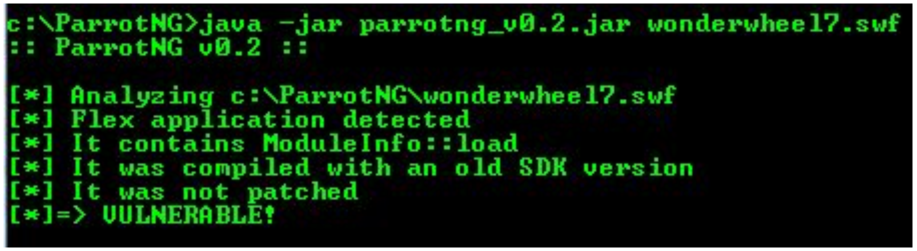
- ParrotNG to the rescue!
  - with Burp Suite extension (passive scan)!
    - Make sure it is working properly -> Important ;-)
    - Only scan .swf extensions!
  - Can search a folder
- Decompile & Search:

```
Security.sandboxType == Security.REMOTE)
```

  - In “mx.modules.ModuleManager”
  - Patched version may have “&& *false* == *true*”
- Cross-Site Content Hijacking (XSCH) PoC :
  - https://github.com/nccgroup/CrossSiteContentHijacking

CVE-2011-2461 Exploitation PoC

- "wonderwheel7.swf" was hosted on Google.com originally
- ParrotNG detected the issue:



```
c:\ParrotNG>java -jar parrotng_v0.2.jar wonderwheel7.swf
:: ParrotNG v0.2 ::

[*] Analyzing c:\ParrotNG\wonderwheel7.swf
[*] Flex application detected
[*] It contains ModuleInfo::load
[*] It was compiled with an old SDK version
[*] It was not patched
[*]=> VULNERABLE!
```

- e.g: Hijacking contents from "0me.me" by "15.rs":
  - https://15.rs/ContentHijacking/ContentHijackingLoader.html?objfile=**https://0me.me/demo/cve-2011-2461/wonderwheel7.swf**&objtype=cve-2011-2461&target=**https://0me.me/secret.txt**&logmode=result&isauto=1

Important: Do Not Reinvent the Wheel!

- Search for known vulnerabilities
  - e.g.: https://web.archive.org/web/20130730223443/http://web.appsec.ws/FlashExploitDatabase.php
- Search their issue tracker for security issues
- Old exploits may still be valid with a few changes!

Automated Testing

Listed in OWASP Flash Security Project:

- FlashDiggity
  - Decompile -> Search using RegEx
  - Extractable Rules: http://www.bishopfox.com/dictionaries/Flash%20Regexes.txt
  - Had problems with AS3 during test
- HP SWFScan (Part of HP WebInspect)
  - Decompile AS2 & 3 -> Search using RegEx
  - Has module exclusion rules
  - Stand-alone is old otherwise commercial
- HP Fortify
  - Scan AS3, Flex3 & 4 using source code (not SWF)
  - Commercial

Updated SWFIntruder +

- Updated SWFIntruder:
  - Dirty fix for the original SWFIntruder
  - Uses several payloads for each input parameter
  - Can detect most of AS2 FlashVars
  - FlashVars should be declared for AS3
  - Good to find XSS without user interaction
  - Runs in a browser – can be slow
  - Can be extended by you! https://github.com/irsdl/updated-SWFIntruder
- FlashBang
  - Runs in a browser
  - Based on Mozilla's Shumway
  - Easy way to identify FlashVars (just has some bugs!)
  - https://github.com/cure53/flashbang

Try it on! Homework!

- http://0me.me/swfintruder/testSWF/
  - http://0me.me/swfintruder/testSWF/clickTagSample.swf
  - http://0me.me/swfintruder/testSWF/fileuploader.swf
  - http://0me.me/swfintruder/testSWF/Vulnerable.swf

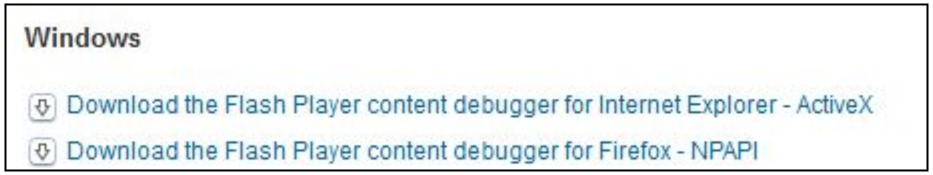
Manual Testing

- Preparing testing environment
- Compiling ActionScript files
- Decompiling SWF files
- Finding inputs (sources)
- Finding usage of dangerous functions (sinks)
- Reviewing the logic, finding sensitive strings, reversing, etc.

Preparing the Environment (Windows)

- Download the Flash debugger version:
  - https://www.adobe.com/support/flashplayer/downloads.html

  Windows:



Windows

Download the Flash Player content debugger for Internet Explorer - ActiveX

Download the Flash Player content debugger for Firefox - NPAPI

- Modify the "mm.cfg" file in %userprofile%
  - e.g. c:\users\myuser\mm.cfg

```
ErrorReportingEnable=1
TraceOutputFileEnable=1
MaxWarnings=50
PolicyFileLog=1
PolicyFileLogAppend=1
# AS3Trace=1 # To see more!
```

  - Default log file location in Windows (policy file is there too):

```
%userprofile%\AppData\Roaming\Macromedia\Flash Player\Logs\flashlog.txt
```

Compiling HelloXSSWorld.as

- Free recommended IDEs:
  - FDT (similar to Eclipse) (preferred for simpler projects)
  - FlashDevelop (includes a powerful runtime debugger)
- + Flex SDK and Java
- Code sample (vulnerable to open redirect and XSS):

```
package {
        import flash.net.navigateToURL;
        import flash.net.URLRequest;
        import flash.display.Sprite;

        public class HelloFlashWorld extends Sprite {
                // User input: HelloFlashWorld.swf?target=foo
                private var url : String = root.loaderInfo.parameters.target;

                public function HelloFlashWorld() {
                        var request : URLRequest = new URLRequest(url);
                        try {
                                // redirect to the target URL
                                navigateToURL(request);
                        } catch (e : Error) {
                                // handle error here
                        }
                }
        }
}
```

Decompiling a SWF File

- Recommended decompiler: JPEXS Free Flash Decompiler
  - Easy to use UI
  - Can edit SWF files
  - Occasional updates
  - Another Java based tool! can be slow and it might crash but still good!

https://www.free-decompiler.com/flash/

https://github.com/jindrapetrik/jpexs-decompiler

Decompiled, Now What?

- AS1/2 or AS3?
  - http://dev.sitedaniel.com/swfinfo/swfinfo.swf - added to Updated SWF Intruder
- Find input parameters (sources)
  - Find their usage
- Find interesting/sensitive functions (sinks)
  - Check their inputs
- Identify insecure policies
  - Any interesting behaviour?
- Identify sensitive data or hidden URLs
  - Can lead to server-side issues (more serious issues)
- Identify storage and logging issues
  - Cookies and logs

Input Parameters - Sources

- Finding a "source":
  - Look at the HTML page (DOM viewer)
  - Find similar inputs based on a known input parameter
  - AS3 (Variables need to be defined):
    - Search for: "*root*", "*loaderInfo*", "*parameters*"

```
\.(root|loaderInfo|parameters)[^\w]|[^\w](root|loaderInfo|parameters)\.
```

    - e.g.: *root.loaderInfo.parameters.inputName*
  - AS2 (Variables can be undefined):
    - Search for: "*_root*", "*_global*", "*_level0*"

```
\.(_root|_global|_level0)[^\w]|[^\w](_root|_global|_level0)\.
```

    - Any undefined variable! Use Flash debugger log file!

```
Warning: Reference to undeclared variable, 'inputName'
```

Sinks

- Sinks - find usage of sensitive functions
  - Can run JavaScript:
    - AS3: "*ExternalInterface.call*", "*navigateToURL*"
    - AS2: "*getURL*", "*fscommand*"
    - "*.htmlText*"
  - Can load objects, or send/receive/store data:
    - "*XMLLoader*", "*AMFService*", "*SWFLoader*", "*loadVariables*", "*loadMovie*", "*loadMovieNum*", "*LoadVars.load*", "*LoadVars.send*", "*NetStream.play*", "*getDefinition*", "*getDefinition*", "*FScrollPane.loadScrollContent*", "*XML.load*", "*Sound.loadSound*", "*NetStream.play*", "*URLRequest*", "*URLLoader*", "*URLStream*", "*LocalConnection*", "*SharedObject*"
  - Can run Flash functions from JavaScript:
    - "ExternalInterface.addCallback" (AS3), ".watch" (AS2)
    - Important with insecure "*Security.allowDomain*"
- No sensitive function = Less likely to find a good vulnerability

Source <-> Sink Flow!

- Tainted source --> … --> sink!
- Sink <-- … <-- Tainted source!

- Any validation?
  - What is allowed?
  - Is it good enough?

- Any logic?
  - Some inputs should be set for something to happen?
  - Role of any provided external file/URL

Insecure Policies in SWF Files

- Search for "*allowDomain*" and "*allowInsecureDomain*"

- *Security.allowDomain:* Cross-domain communication
  - SWF can be scripted by another SWF file on another domain
  - HTML (JavaScript) from another domain can communicate with SWF
- *Security.allowInsecureDomain:* HTTP to HTTPS communication
  - HTTPS communication to HTTP is fine

- *LocalConnection*'s *Security.allowDomain*
  - SWF/AIR can communicate with another SWF/AIR

Not an issue if there is no interesting functionality!

Sensitive Data / Hidden URLs / Gems!

- Think like a forensic analyst! Search for:
  - URLs
  - Emails
  - Secret keys and passwords
  - Database information
  - Etc.
- FlashDiggity rules are good:
  - http://www.bishopfox.com/dictionaries/Flash%20Regexes.txt

Sensitive Data in Storage!

- "SharedObjects" for Flash Cookies!
  - Can even store binary
- "trace" function for logging in debug mode.
  - Can make the debugging easier
  - Sensitive data in log files when debugger version is used

Find More! Be creative!

- Always look at the FlashVars parameter names
  - Anything called “onload”, “onclick”, or “redirect”?
- Does it load another file when you open it? Find it, abuse it!
- Does it accept external configuration files?
  - Find a valid config file and manipulate it
  - Example: XSS issue in FlowPlayer: https://github.com/flowplayer/flash/issues/263

"ExternalInterface.call" XSS Confusion!

- Accept JS function name and its parameters
- Both can lead to XSS
- The first parameter can be a simple JavaScript code (name of JS function)
- The next parameter (argument) is escaped:
  - " turns into \" □ all good!
  - \ doesn't turn into \\ □ too bad!

  So \" can be used to run a JS code. e.g. *\"))-alert('XSS')}catch(e){}//*

  See http://mihai.bazon.net/blog/externalinterface-is-unreliable
- Debuggable using browsers' console – cause an error:
  - *xxx"'(){}\"\'(){}\\'\\"(){}xxx*

Bypassing Client Side Protections

- Protections on the client side only make it more user friendly
  - Not good for security!
- Find the responsible function in the source code
  - Understand how it works, find the credentials, and bypass it!
  - Change the code and save it to bypass the protections

More Issues…

- Identify and review the sensitive functions
  - Such as login or encryption functions
- Flash files can contain unused/commented server side code and information
- Identify requests that it sends to the server
  - Can lead to finding broken access controls on the server side
- Examples:
  - Testing an online game?
    - Can you go to the next level without playing?
  - Does it use encryption?
    - Are there any keys stored in the application?

FlashVars Tips!

- Passing parameters in URL:
  - File.swf?param1=value1&p2=v2
- Removes invalid encoding
  - param1=value1 -> pa%Xram1=val%Yue1
  - param1=value1 -> pa%=ram1=val%#ue1
  - param1=value1 -> pa%AXram1=val%B#ue1
- Sending parameters after “#” is dead? Nope!
  - File.swf?%#param1=value1&p2=v2
- In redirection, %7f-%FF converts to “?”
- BOM (byte-order-mark) “%EF%BB%BF” = a SPACE char!
- Flash in Firefox may not like %00

Examples

- Bypassing firewalls – was detecting “domid=”:
  - https://example.com/foobar/ScrollLine2D.swf?%#domid=\%22))}catch(e){};alert(%27External%20Interface%20XSS%20from:%20%27%2bdocument.domain)//&registerwithjs=1
- Bypassing an in-app protection – didn’t like inputs from GET:

```
pos = root.loaderInfo.url.indexOf('?');
if (pos !== -1) {
    query = parseStr(root.loaderInfo.url.substr(pos + 1));

    for (var key:String in params) {
        if (query.hasOwnProperty(trim(key))) {
            delete params[key];
        }
    }
}
```

  - /flashmediaelement.swf?jsinitfunctio%gn=alert`1`

Demo – Finding Vulnerabilities!

- clickTagSample.swf □ ActionScript2
- vulnerable.swf □ ActionSctipt2

- Homework:
- fileuploader.swf □ ActionScript3
- Answer (in white colour):

- You are ready with more practice!

Used RegExes in Demo

AS3 Inputs:

\.(root|loaderInfo|parameters)[^\w]|[^\w](root|loaderInfo|parameters)\.

AS2 Inputs (remember undefined inputs – follow the sinks):

\.(_root|_global|_level0)[^\w]|[^\w](_root|_global|_level0)\.

XSS:

(getURL|ExternalInterface\.call|navigateToURL|\.htmlText)

Sensitive functions:

(XMLLoader|AMFService|SWFLoader|loadVariables|loadMovie|loadMovieNum|LoadVars\.load|LoadVars\.send|NetStream\.play|getDefinition|getDefinition|FScrollPane\.loadScrollContent|XML\.load|Sound\.loadSound|NetStream\.play|URLRequest|URLLoader|URLStream|LocalConnection|SharedObject)

Interesting keywords:

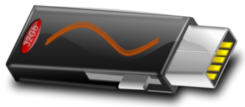
(allowInsecureDomain|allowDomain|ExternalInterface|load|xml|sql|url|flashvar|pass|TextField|encr)

Final Notes

- Search in your proxy logs for "SWF" files!
- JS libraries and plugins can contain Flash files
- Can be slow – don't panic! Plan ahead!
- Review the API references for any security-related functions:
  - AS2: http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/2/
  - AS3: http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/
- The following resource is also recommended for code review:
  - http://www.hpenterprisesecurity.com/vulncat/en/vulncat/index.html
- Flash files can send requests to their server during testing!
- Downloading random Flash files is dangerous but fun
  - We all know why!

Thank you! Questions? Really? Why?! ;)

- Sample files in: https://github.com/irsdl/Flash-Files-Vulnerability-Database



References & Further Reading - 1

- Securely deploying cross-domain policy files
  - http://blogs.adobe.com/security/2009/11/securely_deploying_cross-domai.html
- Related to Flash policy file
  - http://www.adobe.com/devnet/adobe-media-server/articles/cross-domain-xml-for-streaming.html
- Security Domains, Application Domains, and More in ActionScript 3.0
  - http://www.senocular.com/flash/tutorials/contentdomains/
- Penetration testers guide
  - http://www.ivizsecurity.com/blog/web-application-security/testing-flash-applications-pen-tester-guide/
- Exploiting CVE-2011-2461 on google.com
  - http://blog.mindedsecurity.com/2015/03/exploiting-cve-2011-2461-on-googlecom.html
- AS3 hidden treasure in the mm.cfg file
  - https://jpauclair.net/2010/02/10/mmcfg-treasure/

References & Further Reading - 2

- ParrotNG project to find CVE-2011-2461 vulnerable files
  - https://github.com/ikkisoft/ParrotNG
- Testing for Cross site flashing
  - https://www.owasp.org/index.php/Testing_for_Cross_site_flashing_(OTG-CLIENT-008)
- Blinded by Flash: Widespread Security Risks Flash Developers Don't See
  - https://www.blackhat.com/presentations/bh-dc-09/Jagdale/BlackHat-DC-09-Jagdale-Blinded-by-Flash.pdf
- SWF INFO : WIDTH, HEIGHT, SWF VERSION, ACTIONSCRIPT VERSION, FRAMERATE
  - http://blog.sitedaniel.com/2009/11/swf-info-width-height-swf-version-actionscript-version-framerate/
- Creating more secure SWF web applications
  - http://www.adobe.com/devnet/flashplayer/articles/secure_swf_apps.html
- OWASP Flash Security Project
  - https://www.owasp.org/index.php/Category:OWASP_Flash_Security_Project

References & Further Reading - 3

- Same Origin Method Execution (SOME)
  - http://www.benhayak.com/2015/06/same-origin-method-execution-some.html
- WordPress SOME bug in plupload.flash.swf
  - https://gist.github.com/cure53/09a81530a44f6b8173f545accc9ed07e
- Catch-up on Flash XSS exploitation
  - https://soroush.secproject.com/blog/2013/10/catch-up-on-flash-xss-exploitation-bypassing-the-guardians-part-1/
  - https://soroush.secproject.com/blog/2013/10/catch-up-on-flash-xss-exploitation-part-2-navigatetourl-and-jar-protocol/
  - https://soroush.secproject.com/blog/2014/01/catch-up-on-flash-xss-exploitation-part-3-xss-by-embedding-a-flash-file/