Test.Security(Flash);

Lavakumar Kuppan
lava<at>andlabs<dot>org

OWASP

21st March, 2009

The OWASP Foundation

http://www.owasp.org

About me:

- Have been doing security auditing for 3 years
- Performed more than 100 penetration tests
- Perl and C# programmer

I write code for pleasure

And break code at work ☺

Imagine you are testing a web application and it has flash content.
What would you do?

a) Ignore the flash content
b) Enjoy the flash videos and get back to testing the rest of the application
c) Badmouth the developers for using silly programs like flash in a serious application
d) Include the flash app in your test scope

If you answer is a/b/c then listen carefully…

Agenda

Introduce the bare minimum that every developer and Penetration tester should know about flash security

What is Flash

- Multimedia platform from Adobe(Macromedia)
- Ideal for animations and graphics
- Files have .swf extension
- Flash embedded in web pages is played by the Flash player plug-in of the browser

The moment of truth

Who hasn't played flash games at work????

Logic in Flash

- Logic can be built in to Flash applications with **ACTIONSCRIPT**
- Actionscript is the programming language for Flash applications
- This is what powers your favorite flash game.
- ActionScript 3.0 is the latest version.
- We will talk about ActionScript 2.0, its more widely used

Flash is powerful

Flash applications can:

- Send HTTP requests to third-party domains
- Create XML socket connections
- Store data on the file system permanently (LocalSharedObjects)
- Can access the DOM of the page
- Execute JavaScript

Areas of focus

- Cross-site scripting in flash
- Cross-site Flashing
- Crossdomain concerns
- Sensitive data storage
- Encryption in flash
- Attacking the server

_global and _root objects

- Attributes of _global and _root objects are represented as:
  - _root.variableName
  - _global.variableName
- If these are undefined then they can be initialized from the querystring

Example:

Actionscript source of demo.swf:

```
class Demo {
	static var app : Demo;
	function Demo() {
		if (_root.url != undefined )
		{
			getURL(_root.url);
		}
	}
	// entry point
	static function main(mc) {
		app = new Demo();
	}
}
```

In this example the value of _root.url can be initalised from the querystring:

http://10.10.10.10/flash/demo.swf?url=http://www.owasp.org

Cross-site scripting

- In the previous example if the user enters javascript:alert(1) as the URL then script execution is possible

User enters:

http://10.10.10.10/flash/demo.swf?url=javascript:alert(1)//

Passed to the getURL function:

getURL(javascript:alert(1)//); - Cross-site scripting

Vulnerable functions a.k.a PDNF

- **loadMovie()**
- **getURL()**
- **loadMovie()**
- **loadMovieNum()**
- **FScrollPane.loadScrollContent()**
- **LoadVars.load()**
- **LoadVars.send()**
- **XML.load ()**
- **LoadVars.load ( )**
- **Sound.loadSound( );**
- **NetStream.play();**

- All these functions take URL as an input parameter.
- To exploit, inject the URL parameter with:
  asfunction:getURL,javascript:evilcode
- Eg:
  http://victim/file.swf?URL=asfunction:getURL,javascript:evilcode

Other means of Cross-site scripting

- Text fields in flash can be injected with HTML
  - textfield.html = true
  - textfield.htmlText = ’<a href='javascript:alert(1)' >’
- flash.external.ExternalInterface.call();
  - This function can call JavaScript methods.
  - Method Description:

public static call(methodName:String, [parameter1:Object]) : Object

Cross-site flashing (XSF)

According to the OWASP testing guide:

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 to command an Adobe 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.

Could lead to leakage of data or manipulation of the normal functioning of the flash file.

Cross Domain concerns

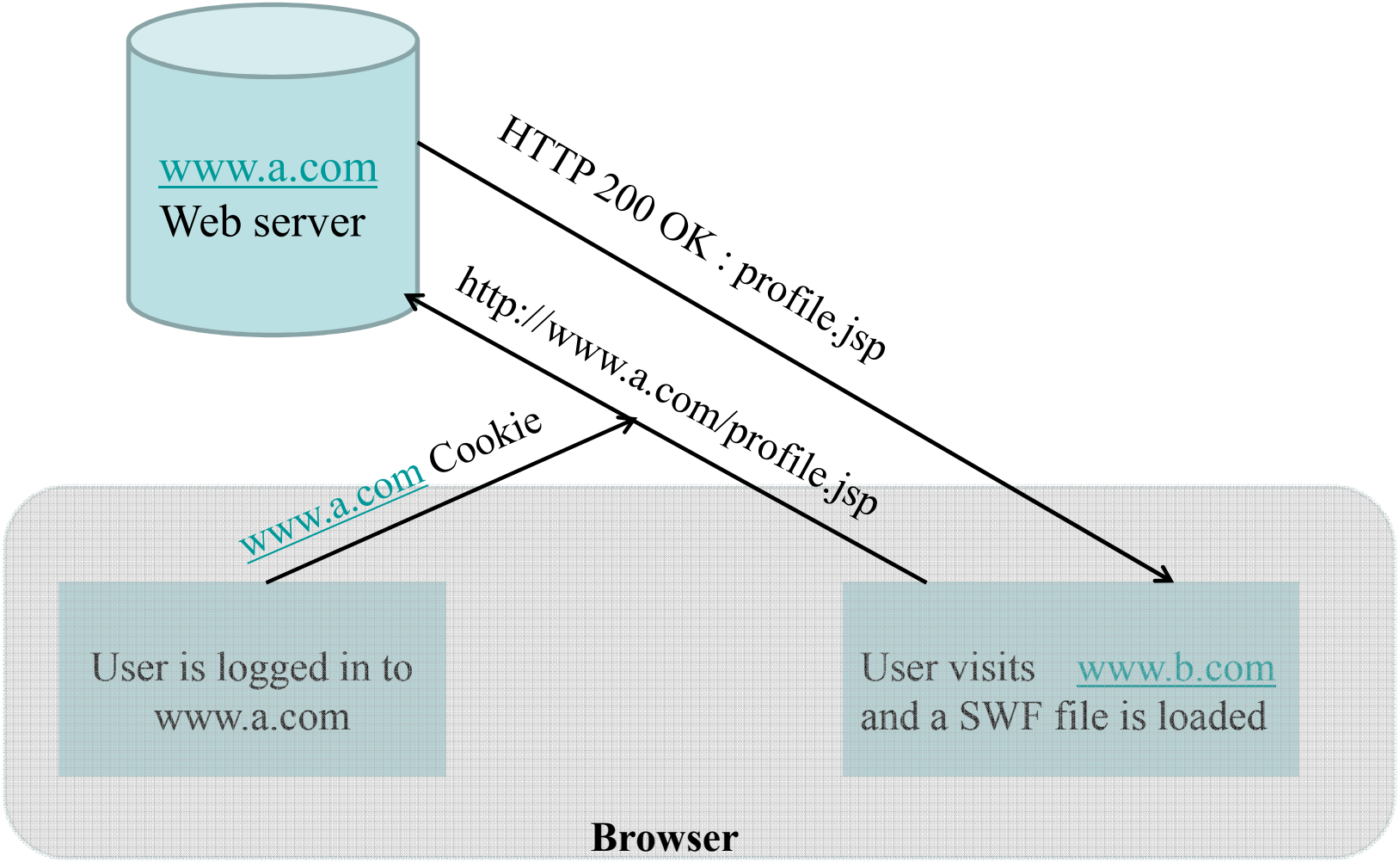
- Crossdomain.xml
- Allowscriptaccess
- Localconnection
- security.allowDomain()

Crossdomain.xml

- It's a policy file that allows SWF files from external domains to make HTTP calls to this domain
- Sample Crossdomain.xml file:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "/xml/dtds/cross-domain-policy.dtd">
<!-- Policy file for mysite.com -->
<cross-domain-policy>
    <!-- This is a master-policy file -->
    <site-control permitted-cross-domain-policies="master-only"/>
    <allow-access-from domain="www.example.com" secure="true" />
    <allow-access-from domain="*.example.com"  />
</cross-domain-policy>
```

How it works


www.a.com
Web server
HTTP 200 OK : profile.jsp
http://www.a.com/profile.jsp
www.a.com Cookie
User is logged in to www.a.com
User visits www.b.com and a SWF file is loaded
Browser

Crossdomain.xml

- Sites relying on cookies for session management should be careful about allow external sites
- Never use the universal allow wildcard - * - **Dangerous!!**
- Even sites on the intranet should have strict crossdomain.xml files
- Secure attribute should always be set for HTTPS content
- Permitted-cross-domain-policies should always be set to 'master-only'

Allowscript access

- Embedding swf files in HTML

```
<object id='MyMovie.swf' classid='clsid:D27CDB6E-AE6D-11cf-96B8-
    444553540000'
    codebase='http://download.adobe.com/pub/shockwave/cabs/flash/swflash.
    cab#version=9,0,0,0' height='100%' width='100%'>
<param name='allowScriptAccess' value='sameDomain'/>
<param name='src' value="MyMovie.swf'/>
<embed name='MyMovie.swf' pluginspage='/go/getflashplayer'
    src='MyMovie.swf' height='100%' width='100%'
    allowScriptAccess='sameDomain'/>
</object>
```

What it does

- The value of this setting determines the script access to the SWF
- Possible values:
  - never – No script access allowed.(Deprecated)
  - sameDomain – SWF from same domain have script access
  - always – SWFs from external domains also have script access – **Dangerous!!**

Localconnection

- Used for interprocess communication between flash files
- One flash file can call methods in another flash file using this even if they are from different domains
- Access control is enforced using the LocalConnection.allowDomain() method
- LocalConnection.allowDomain('*') allows SWF files from all domain – **Dangerous!!**

Security.allowDomain()

- Normally SWF loaded from www.a.com cannot access the variable, objects, properties and methods of SWF loaded from www.b.com
- But Security.allowDomain() can be used to bypass this security restriction.
- HTTP to HTTPS restriction can be overcome using System.security.allowInsecureDomain() - **Dangerous!!**
- System.security.allowDomain("*") – **Dangerous!!**

Sensitive data storage

- Any hard-coded password or other sensitive information in the SWF file is a major risk
- SWF files can be decompiled easily
- SharedLocalObjects are like cookies in flash
- They are used to store information on the client-side
- This information is stored in clear-text
- HTTP to HTTPS access is restricted with 'secure' flag

var mySO = SharedObject.getLocal("userInfo", null, **false**);

- **Dangerous!!**

Flash decompilation with Flare

- Insecure.as

```
class Demo {
        static var app : Demo;

        function Demo() {
                var username = "administrator";
                var password = "p@ssw0rd";
//------------------------cut here ------------------------
```

Decompiling with flare.exe: C:\>flare.exe insecure.swf

- Insecure.flr

```
movie 'talk.swf' {
// flash 7, total frames: 1, frame rate: 20 fps, 800x600 px,
compressed

  movieClip 20480 __Packages.Demo {

    #initclip
      if (!Demo) {
        _global.Demo = function () {
          var v2 = 'administrator';
          var v3 = 'p@ssw0rd';
        };

//------------------------cut here ------------------------
```

Encryption in Flash

- Any attempts at client-side encryption is a bad idea
- Key has to be hard-coded and can be stolen
- Even if the SWF file uses HTTPS, serving the file over HTTP is very **dumb!!**
- Best way to ensure data security is to serve the SWF file over HTTPS
- If you see anything else happening then its surely a broken security model

Attacking the server

- There could be two-way communication between SWF file and the server
- Data form the SWF file could be used in SQL queries or other potentially dangerous system commands
- Since the data is coming from the SWF file, developers tend to consider it to be safe and fail to validate it properly
- Identify data sent to the server and fuzz them for common injection vulnerabilties

Credits, References and further reading

- OWASP testing guide V3 http://www.owasp.org/images/5/56/OWASP_Testing_Guide_v3.pdf
- Creating more secure SWF web apps http://www.adobe.com/devnet/flashplayer/articles/secure_swf_apps.html
- https://www.flashsec.org/
- Stefano Di Paola, **OWASP Flash Security Project**
- **Flare,** http://www.nowrap.de/flare.html

Thank You

?