Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Príprava štúdia matematiky a informatiky na FMFI UK v anglickom jazyku

ITMS: 26140230008

dopytovo – orientovaný projekt

Moderné vzdelávanie pre vedomostnú spoločnosť/Projekt je spolufinancovaný zo zdrojov EÚ

Rich internet applications: security of Flash and HTML5

Bezpečnosť IT infraštruktúry

Michal Rjaško

2012/2013

Flash vs. HTML
Before HTML5

- Flash is popular because of poor functionality of HTML environment.
  - It allows to add interactive content to websites (video, animations, rich applications)
- Inconsistent HTML among browsers – developers need to test their webs in every browser
- Poor support w3c standards among browsers (mainly in the most popular IE <= 8)

Flash

- Multiplatform multimedia platform ☺
- Ideal for animations, graphics, videos and simple games
- Flash file can be embedded into a web page, where it is played by a browser plugin - Flash player
  - Flash Player is third-party plugin (made by Adobe)
- Logic of flash applications and games is based on language ActionScript (currently in version 3.0)
  - ActionScript is similar to JavaScript, but contains types, classes, …

History of Flash

- 1993 - SmartSketch (Jonathan Gay, FutureWare software)
  - Vector graphic editor for Pen PC – computer controlled by pen
- 1995 – FutureSplash Animator (used by MSN and Disney)
- 1996 – Macromedia Flash 1.0
- 2005 – Adobe Flash 8.0

History of Flash

Flash 2	1997	Buttons, libraries, stereo audio, improved bitmap integration
Flash 3	1998	Alpha transparency, MP3
Flash 4	1999	Streaming MP3 (preinstalled in IE 5)
Flash 5	2000	ActionScript
Flash 6	2002	Video
Flash 7	2003	Charts & graphs, text effects, ActionScript 2.0
Flash 8	2005	Gif & Png, significant new video codec
Flash 9	2006-2007	ActionScript 3, AVM2, FullScreen, H.264, AAC, Flex Builder 2
Flash 10	2008	3D, advanced text features, AIR
	2010-2011	MultiTouch, dynamic streaming, hardware decoding H.264, 64-bit, Acoustic Echo Cancellation, Flash for Mobile Devices
Flash 11	2011-2013	3D hardware, secure random number generation, protected dynamic streaming, workers, ...
Flash 12	2014	Mainly focused on mobile application development

Flash is strong platform

It can:

- Send HTTP requests to a different (than original) domain
- Create socket connections
- Store data on client computer (SharedObjects)
- Access camera, microphone
- Access DOM of its web page
- Execute JavaScript
- Load other Flash files

With emerging HTML5
the future of Flash is uncertain

- Apple refused to support Flash in mobile devices
- Since version 11.1 Adobe does not develop Flash player for mobile devices
- Versions from 11.2 - plugin for linux only via „Pepper“ API, currently the only supported browser in linux is Google Chrome
- Primary orientation on „Premium Video“ and Games
  - Adobe AIR, mobile applications

HTML5 - history

“It must be admitted that many aspects of HTML appear at first glance to be nonsensical and inconsistent.”

“HTML, its supporting DOM APIs, as well as many of its supporting technologies, have been developed over a period of several decades by a wide array of people with different priorities who, in many cases, did not know of each other's existence.”

[w3.org/TR/html5/introduction.html#introduction]

HTML5 - history

- **1990-1993** :: first versions derived from SGML, utilized by CERN
- **1995** :: W3C released HTML 3.0
- **1997** :: HTML 3.2 – many new features
- **1998** :: HTML 4 – used till now. DOM level 1, W3C decided to develop XHTML
- **2000** :: DOM level 2 - getElementById(), events
- **2000** :: W3C released XHTML 1.0,
  – development of XHTML2 started
- **2004** :: DOM level 3

HTML5 - history

- **2004** :: Idea of HTML5 was born, WHATWG founded
  - W3C is not participating, but continues to develop XHTML2
- **2005** :: AJAX, XMLHttpRequest
- **2006 – 2007** :: W3C redecided, now participating in HTML5
- **2007 – now** :: WHATWG and W3C cooperate on standardization of HTML5
- **2012** :: HTML5 W3C Candidate Recommendation
- **End of 2014** :: HTML5 W3C Recommendation

HTML5 - history

- **In July 2012**, WHATWG and W3C decided on a degree of separation
  - W3C focuses on specification of a single definitive standard – “snapshot” of WHATWG
  - WHATWG will continue on HTML5 as “living standard” – features can be added but not removed

HTML5 – Current status

- HTML5 is not ready yet! – work in progress
  - Current status „Candidate recommendation“
    - No fundamental changes are expected
  - Next phases: Proposed Recommendation, Recommendation
  - Recommendation planned for the end of 2014
- W3C != WHATWG, HTML5 != HTML5
- Incomplete support among browsers
  - Old browsers are still in the game (IE8, IE7, ...)

HTML5 – Current status

- W3C specification has about 4.4 MB of text
- WHATWG specification – 707 A4 pages
- This is a lot of implementation work
- Don’t forget about
  - CSS3
  - JavaScript
  - SVG, MathML
  - Canvas, etc. etc.

HTML5 - Security

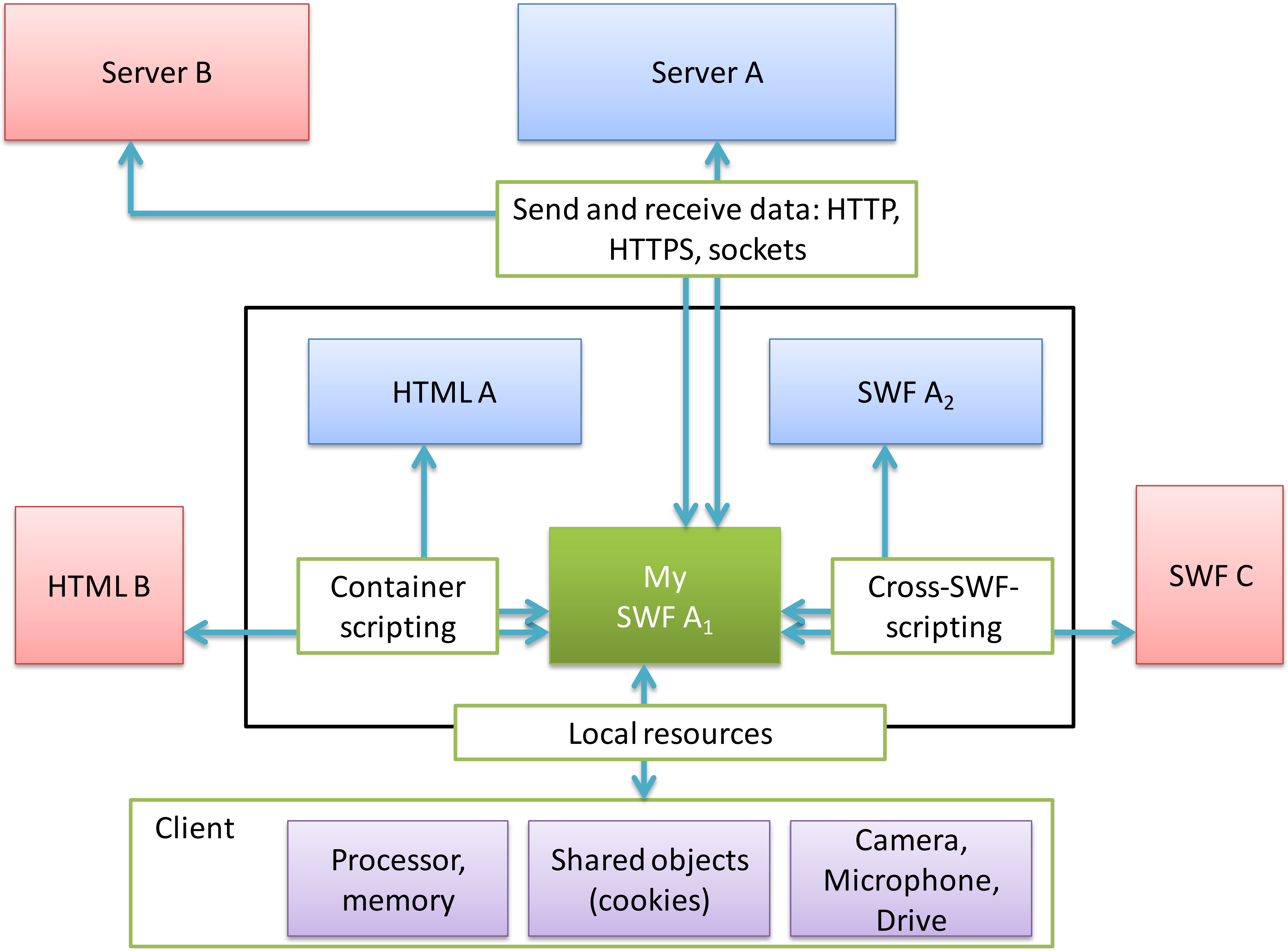
- Some say HTML5 itself is a vulnerability
- Secure implementations require:
  - Clear specifications ✗
  - Manageable amount or work ✗
  - Thorough and diverse testing ✗
  - Fast and precise feedback loops ✗
  - Quick and comprehensive patch deployment ✗

HTML5 – Security

- Inconsistent and still evolving specs
- Browsers rush for implementation
- Web developers build still buggy websites
- Necessary legacy support
  - IE7, IE8, IE6 are still around … government, schools, …

HTML5 vs. Flash

FLASH


Server B
Server A
Send and receive data: HTTP, HTTPS, sockets
HTML A
SWF A_2
HTML B
Container scripting
My SWF A_1
Cross-SWF-scripting
SWF C
Local resources
Client
Processor, memory
Shared objects (cookies)
Camera, Microphone, Drive

Threats

- Bugs in Flash player – AVM
- Decompilation
- Cross-site scripting, Cross-site flashing
- Communication with server, bad cross-domain policy, DDOS attacks
- Phishing
- Access to local resources (Clipboard, Camera, Microphone, Harddrive, FullScreen)
- Utilization of processor and memory

Bugs in Flash player – AVM

- Source code is compiled into byte code
- Flash player executes byte code via AVM
  - In case of AVM2 its possible that AVM compiles byte code to machine code (JIT compilation).
- 4 phases: loading, linking, verification, execution
  - Phases are overlapping, verification occurs in every phase
- Bugs and errors in verification process enable attackers to execute malicious code

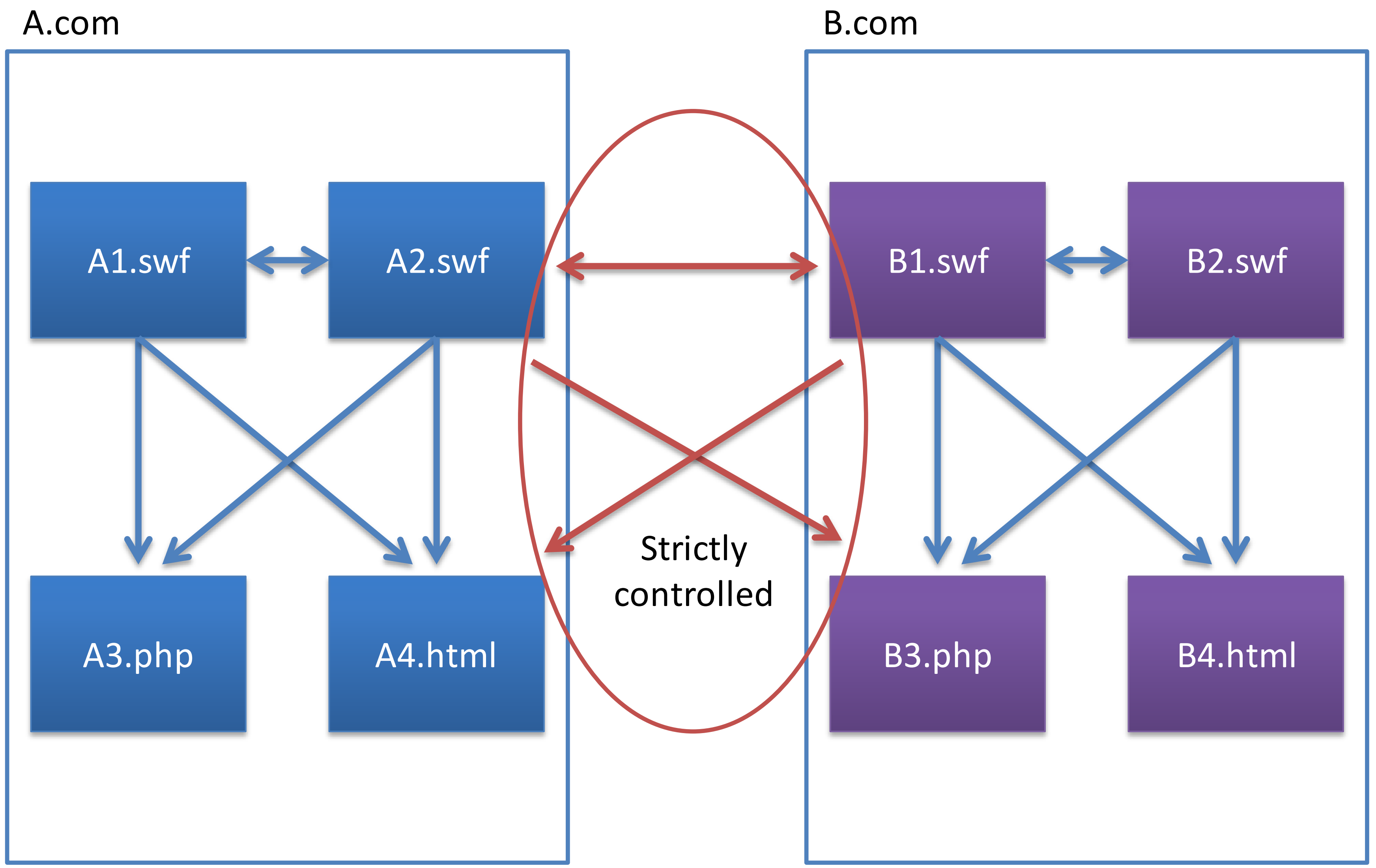
Decompilation

- Specification of AVM2 is open
- Bytecode for AVM2 can easily be decompiled
- Attacker can see source code of your application
  - SWF file cannot contain sensitive data
  - Such as encryption/authentication with keys embedded into SWF file
- Phishing attack by decompilation and modification of application

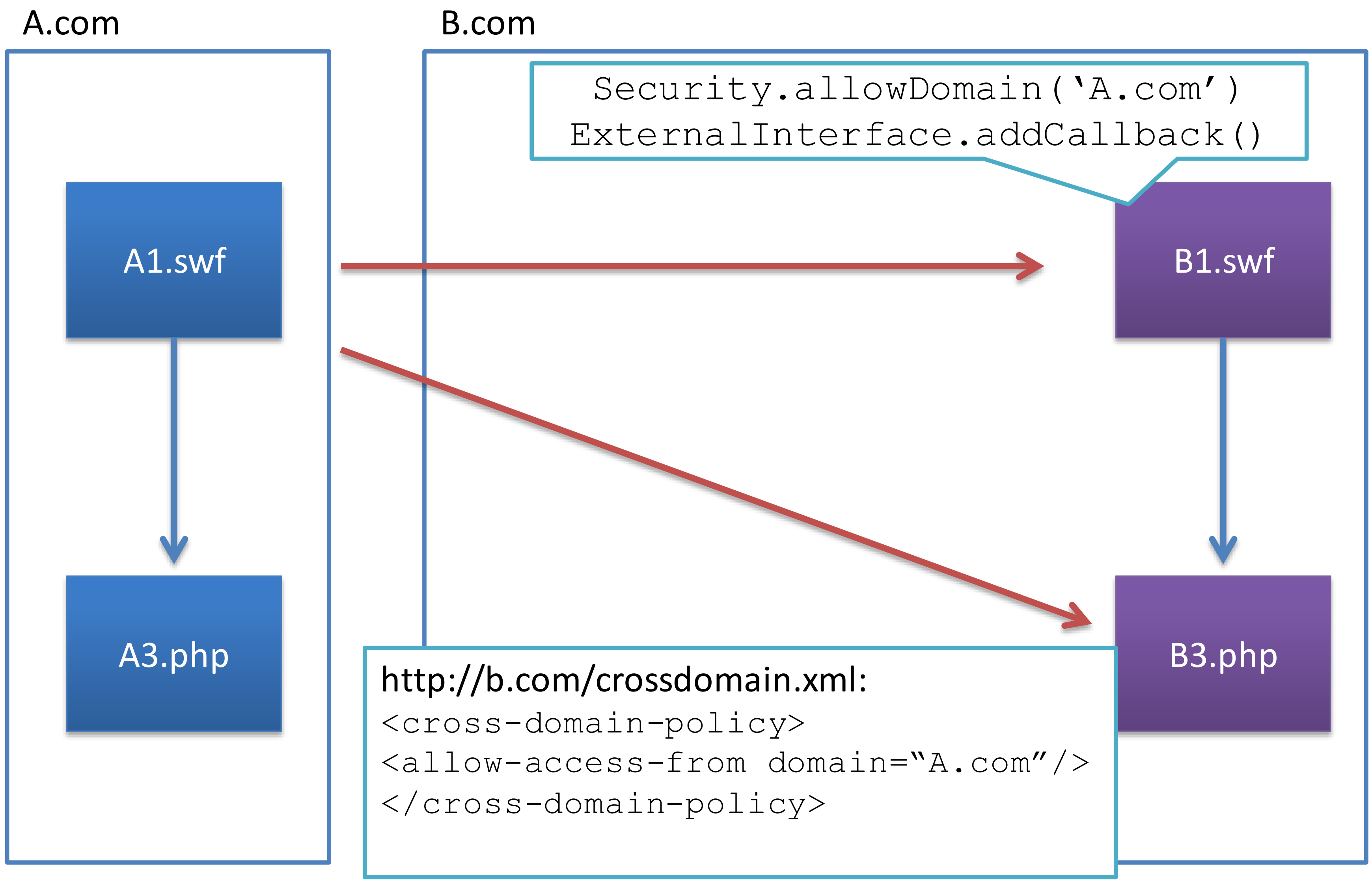
Flash Sandbox model

- Flash assigns to SWF file “sandbox“ based on the domain from which the file is downloaded.
- Communication in the same sandbox is not restricted
- Communication between different sandboxes (domains) is strictly controlled

Flash Sandbox model



Flash Sandbox model



```
Security.allowDomain('A.com')
ExternalInterface.addCallback()
```

http://b.com/crossdomain.xml:

```
<cross-domain-policy>
<allow-access-from domain="A.com"/>
</cross-domain-policy>
```

Communication with server over HTTP
crossdomain.xml

- SWF file can communicate with server in the same sandbox (domain) without any restrictions.
- Request outside the sandbox are controlled using „cross domain policy file“:

```
<cross-domain-policy>
<allow-access-from domain="A.com" secure="true"/>
<site-control permitted-cross-domain-policies="master-only">
</cross-domain-policy>
```

- SWF file can load a non-standard cross domain file (i.e. different from domain.com/crossdomain.xml) by calling:

```
Security.loadPolicyFile("http://B.com/crossdomain.xml")
```

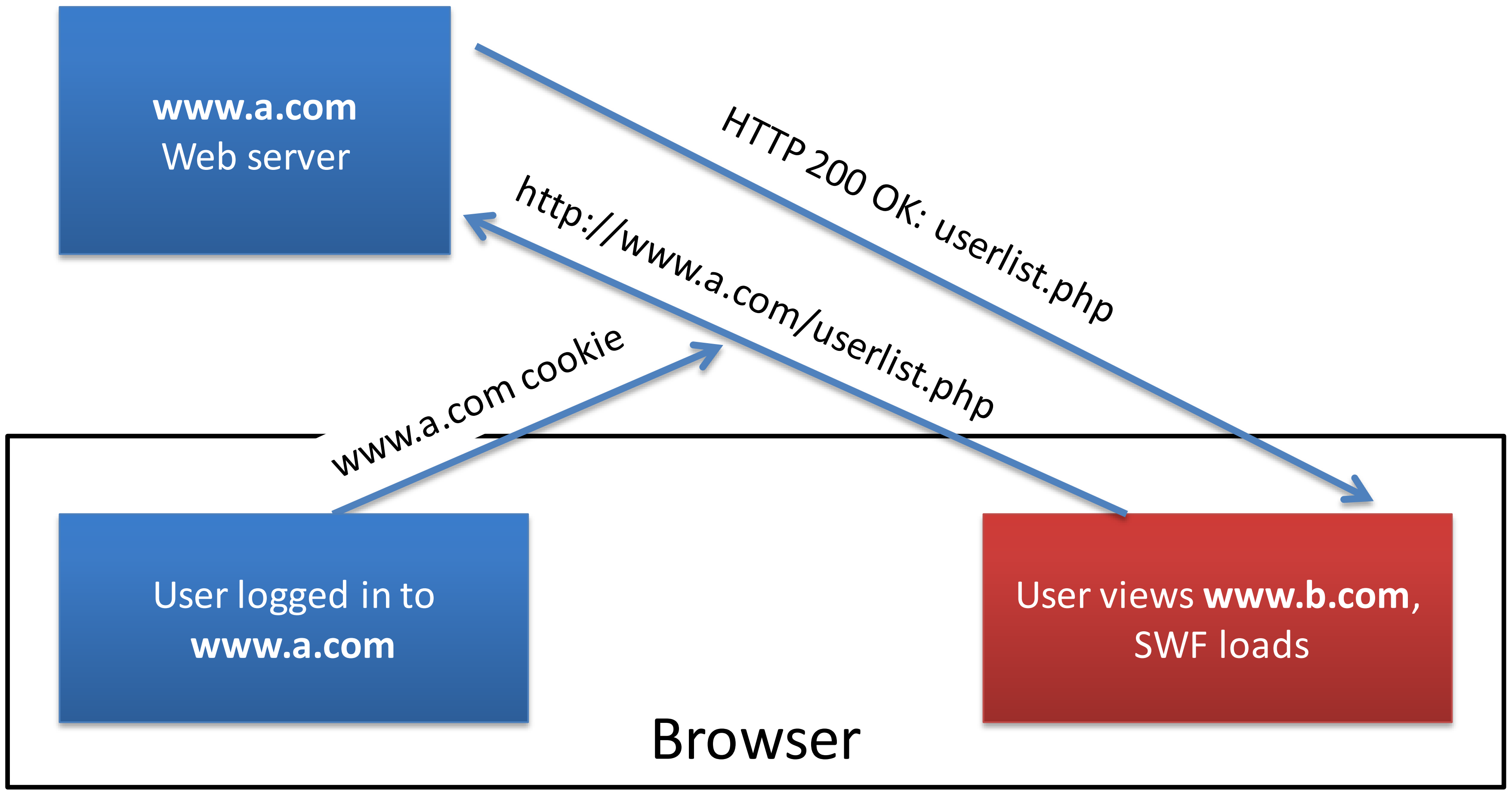
Communication with server – sockets

- Flash supports creating of socket connections (i.e. outside HTTP protocl)
- Same policies as in case of HTTP – cross domain policy file
  - `allow-access-from` can contain ports `ports="3045,4056"`

Communication with server

Some notes

```
<allow-access-from domain="*">
```



www.a.com
Web server
HTTP 200 OK: userlist.php
http://www.a.com/userlist.php
www.a.com cookie
User logged in to **www.a.com**
User views **www.b.com**, SWF loads
Browser

Communication with server
Some notes

```
<allow-access-from domain="*">
```

- DDOS attack☺:
  - MOSAD agent publishes an advertisement on facebook / gmail / …
  - The advertisment is a swf file, which starts sending requests to http://www.elections.ir during elections in Iran

Communication with server
Some notes

```
<site-control permitted-cross-domain-policies=“all”>
```

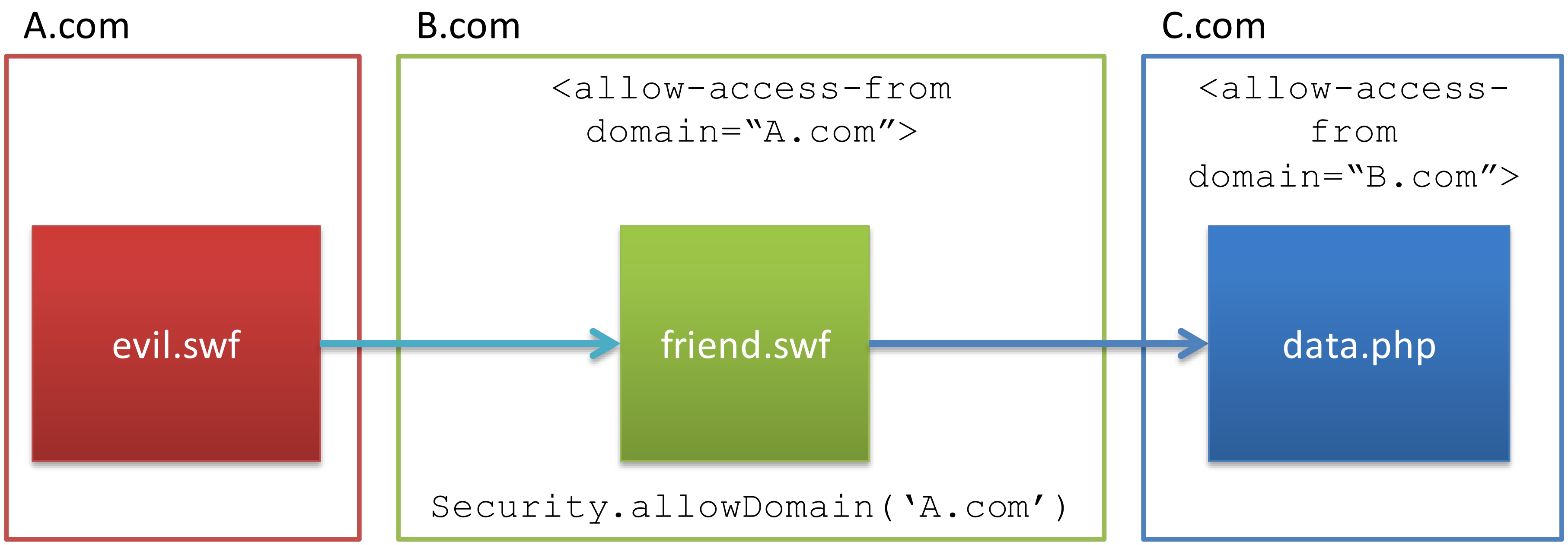
- Attacker can upload his own cross domain policy file (e.g. via CMS).

```
<allow-access-from domain=“A.com” secure=“false”/>
```

- Possible transfer of data over unsecure HTTP protocol (even if the SWF file was loaded via HTTPS).

Communication with server
Some notes

- How to obtain access to an inaccessible domain


A.com
B.com
C.com
<allow-access-from domain="A.com">
<allow-access-from domain="B.com">
evil.swf
friend.swf
data.php
Security.allowDomain('A.com')

Cross-site scripting
attribute allowScriptAccess

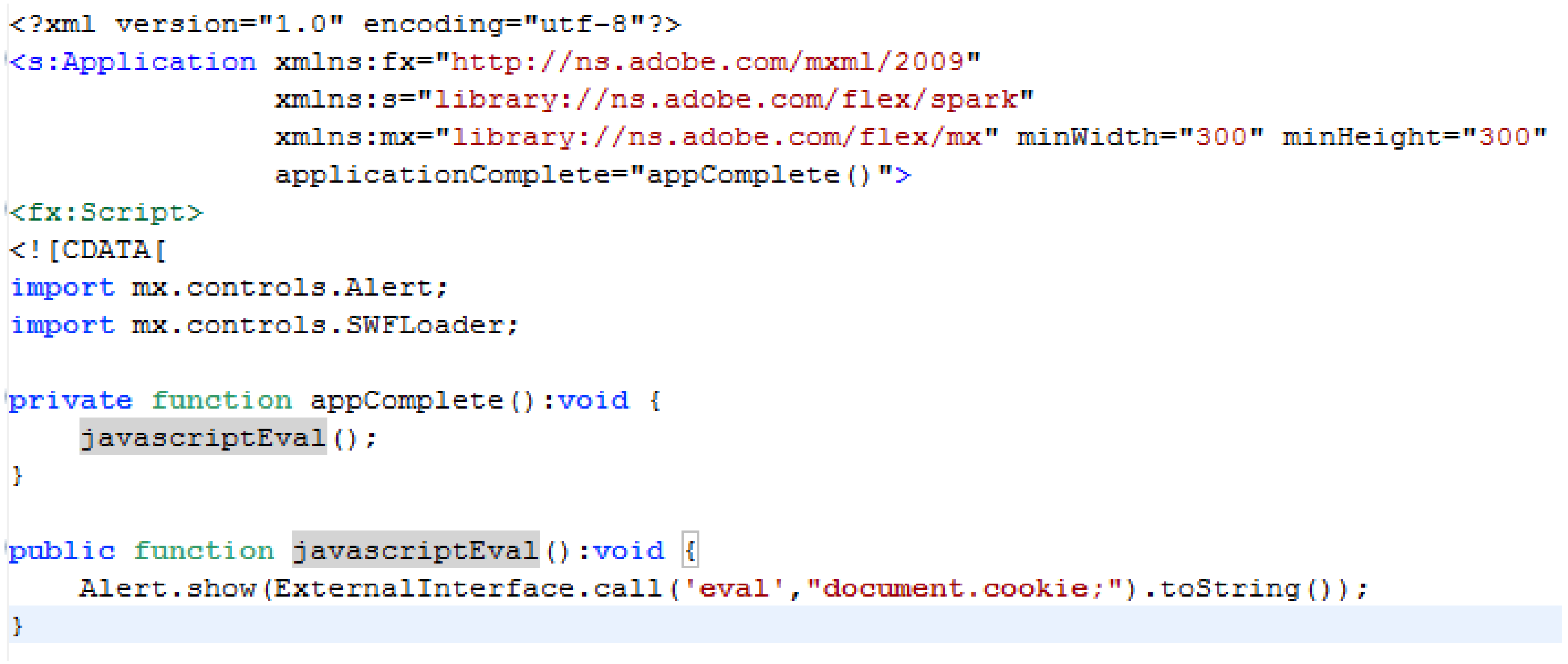
- Embedding of SWF file in HTML:

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
    codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,0,0"
    width="100%" height="100%">
 <param name="src" value="mymovie.swf">
 <param name="allowScriptAccess" value="sameDomain">
 <embed src="mymovie.swf" width="100%" height="100%"
    allowScriptAccess="sameDomain"
    type="application/x-shockwave-flash"
    pluginspage="http://www.macromedia.com/go/getflashplayer">
</object>
```

Cross-site scripting
allowScriptAccess

- Using attribute allowScriptAccess we can restrict access of SWF file to JavaScript
- Possible values:
  - **never** – no acces to JavaScript / DOM
  - **sameDomain** – SWF file from the same domain can access JavaScript / DOM of the web page
  - **always** – SWF files from any domain can access JavaScript - **dangerous!**

Cross-site scripting
allowScriptAccess=“always”



```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
               xmlns:s="library://ns.adobe.com/flex/spark"
               xmlns:mx="library://ns.adobe.com/flex/mx" minWidth="300" minHeight="300"
               applicationComplete="appComplete()">
<fx:Script>
<![CDATA[
import mx.controls.Alert;
import mx.controls.SWFLoader;

private function appComplete():void {
    javascriptEval();
}

public function javascriptEval():void {
    Alert.show(ExternalInterface.call('eval',"document.cookie;").toString());
}
```

Cross-site scripting

FlashVars

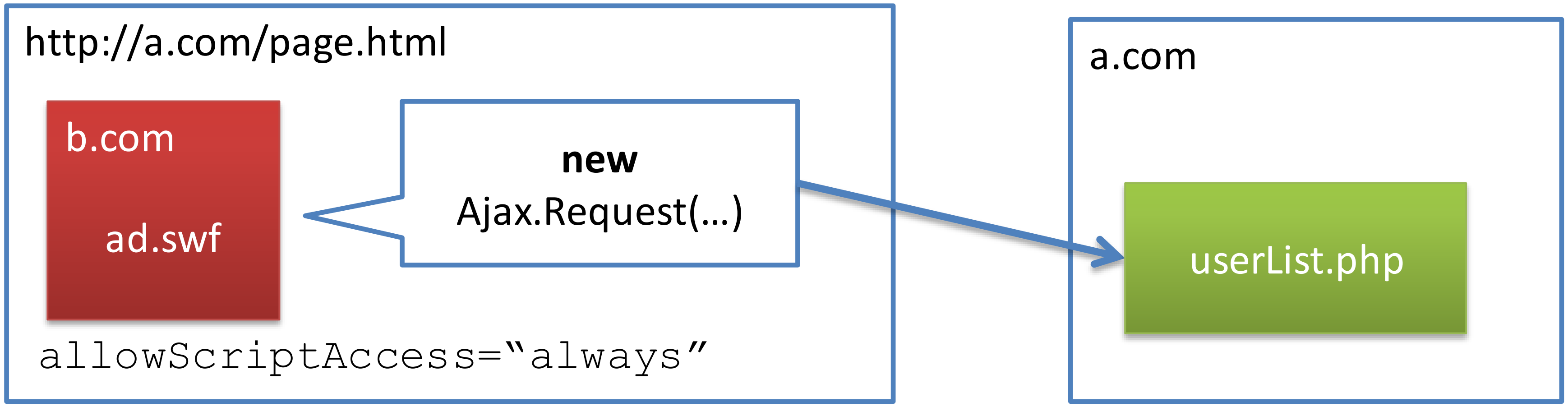
- FlashVars are used for sending data from HTML to Flash player:

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
    codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,0,0"
    width="100%" height="100%">
 <param name="src"
    value="myad.swf?clickthru=http://www.sme.sk">
 <param name="allowScriptAccess"  value="sameDomain">
<param name="flashVars"
      value="clickthru=http://www.sme.sk">
```

- It’s possible to execute javascript using FlashVars
  - mymovie.swf?clickthru=javascript:alert(document.cookie);

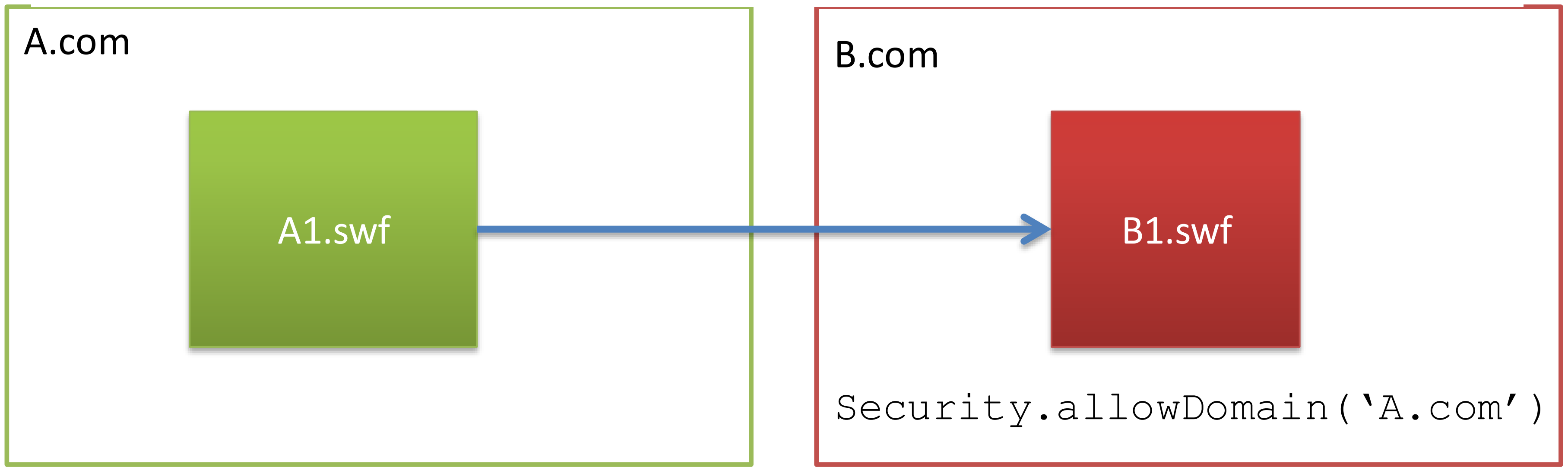
Cross-site scripting
XSS tunneling

- HTTP tunel over XSS channels.
  - SWF file can download data using JavaScript calls (otherwise inaccessible for attacker)
    - E.g. data protected by IP verification, VPN or classical authentication



Cross-Site flashing

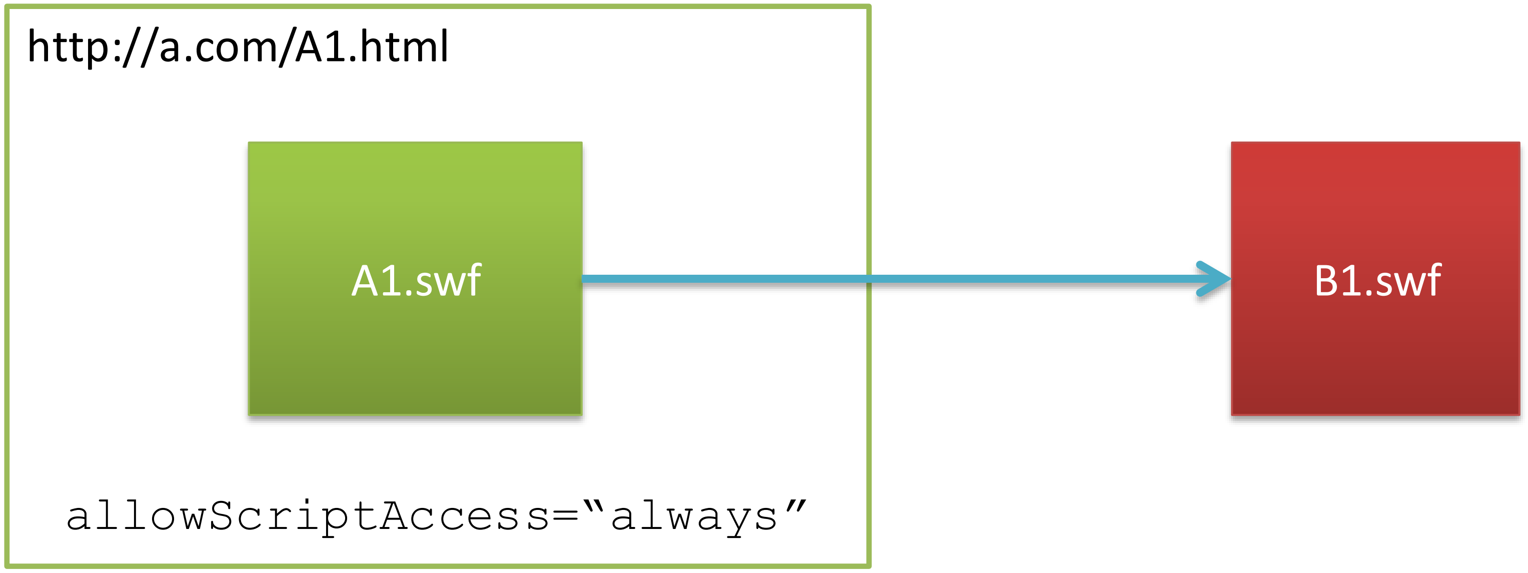
- Flash can launch another flash
  - Function `Security.allowDomain('*')`
  - A1.swf can access methods and properties of B1.swf and vice-versa
  - Possible data exploitation or manipulation with functionality of the SWF file



Cross-Site flashing

Some notes

- allowScriptAccess="always"
- If A1.swf loads B1.swf, B1.swf has access to JavaScript of A1.html
- Possible XSS attack



Cross-Site flashing

Some notes

Two ways how to load another SWF file:

```
var swfloader:SWFLoader = new SWFLoader();
swfloader.load('http://b.com/b1.swf');
```

versus

```
var r:URLRequest = new URLRequest('http://b.com/b1.swf')
var loader:URLLoader = new URLLoader(request);
loader.dataFormat = URLLoaderDataFormat.BINARY;
loader.load(request);
...
swfloader.load(loader.data);
```

In the second case the loaded SWF file has the same sandbox as the loader

Access to local resources

- Proccessor:
  - Flash player allows to terminate script after 15s of execution
  - Till version 11.4 flash application runs in single thread
  - Multiple “Flash” workers supported since 11.4

- Memory:
  - Flash player does not restrict amount of memory used by Flash application

Access to local resources

- Hard drive:

  SharedObjects – cookies for Flash.

    - Maximal size of object per domain is 100 KB, user can change it

  – Flash application can upload/read local files (also multiple files at once)

    - Only by using FileReference class
    - FileReference.browse opens system dialog for file selection
    - The method can only be called in „user-event handler”, i.e. when handling mouse-click or key-press
    - FileReference.download opens system “save file” dialog

Access to local resources

- Clipboard
  - Storing data to clipboard possible only in “user-event handler”
    - Clipboard.generalClipboard.setData()
  - Reading data possible only in “paste” event handler (i.e. when user presses CTRL+V)
    - Clipboard.generalClipboard.getData()
- Camera, Microphone
  - Flash player asks for permission
  - Access can be granted / restricted in Flash player settings

Access to local resources

- Fullscreen
  - Enabled since version 9
  - Key ESC always ends fullscreen mode
  - Fullscreen can be launched only in “user-event” handler
- Access to keyboard is limited in fullscreen mode (DEMO)
  - Till version 10 all keys were restricted
  - Since version 10 only “non-printing” keys are allowed – arrows, space, tab, enter …
  - Since version 11.3 all keys are allowed – special attribute allowFullScreenInteractive
    - Flash player asks for permission

Flash: summary

It's necessary to:

- Limit access of Flash to JavaScript when loading external SWF files
- Take care of JavaScript in FlashVars
- Take care of scripting between SWF-SWF from different domains
- Cross-domain policy file

Flash vs. HTML5

HTML5

HTML5

- Main goal:
  - Create a simple platform for creating interactive web applications
  - Less XML strictness (compared to XHTML), more freedom
  - Emphasis on security

- HTML5 = HTML + JS + CSS (+ SVG + SQL + …)

HTML5 – New features

- New form elements
  - date, tel, color, number, email, url,...
  - Autofocus
  - Form element outside form
  - Validation on client-side
- New attributes Iframe: sandbox, seemless
- History API
- Local storage
- SQLLite – database on client

HTML5 – New features

- Geolocation
- Notifications
- SVG, Canvas
- MathML
- Animations and transformations
- WebGL – 3D acceleration in browser
- Audio / Video
- Webfonts
- Offline application cache
- More semantics:
  - nav, figure, section, …
- CORS – Cross Origin Resource Sharing
- WebSockets
- WebWorkers
- HSTS: HTTP Strict-Transport-Security
- CSP: Content Security Policy
- …

HTML5 – New features

- HTML5 contains several security improvements, but
- In general, new features of HTML5 make attacks easier
  - Great number of new features – new attack scenarios will be developed in future
- It's definitely easier to
  - Track users (Geolocation, localStorage, history API)
  - Cross-site scripting – XSS (CORS, autofocus, new form elements, …)

JavaScript and its security

- Security from 1995
- Two main security requirements
  - Restrict malicious websites to access your computer
  - Restrict malicious website to access another website
- However, if you have all your documents in cloud, who cares about attacker not being able to access your „My documents“ folder?

JavaScript

- Same Origin Policy: scripts on web page cannot communicate with pages in different domain
  - e.g. script loaded into www.fmph.uniba.sk cannot communicate with www.virus.com
- However, scripts loaded into the same page can interact with each other (event from different domains)
  - JavaScript is inherently global
  - Scripts can modify global variables, functions, objects etc. of other scripts.

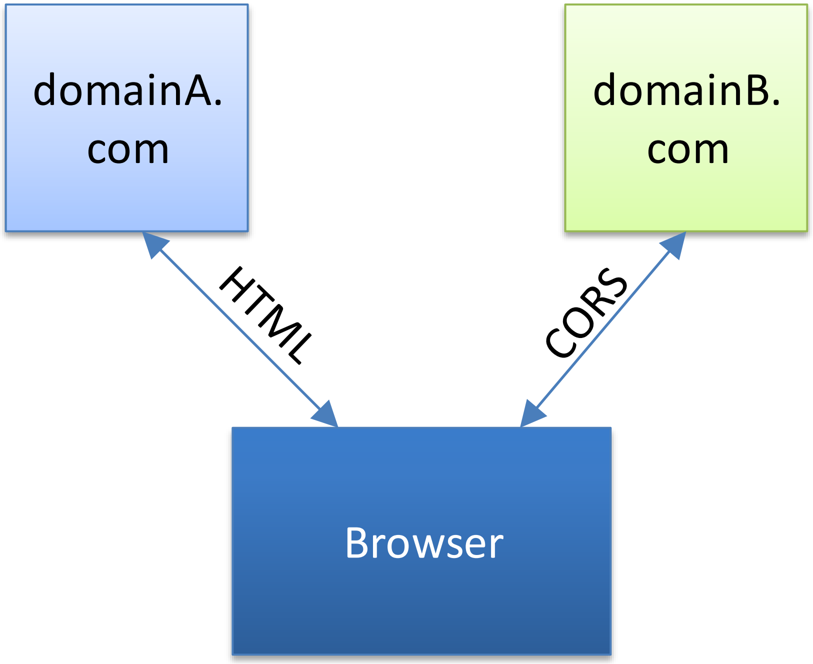
Cross Origin Resource Sharing

- New feature of HTML5
- Enables JavaScript to access websites from different domain
- Similar priciple to crossdomain.xml in Flash
- Based on the HTTP header, browser decides whether to block the given Ajax request or not
  - Compared to crossdomain.xml, CORS has a drawback - flash player loads cross domain policy before the request and blocks the request without contacting the server

Cross Origin Resource Sharing

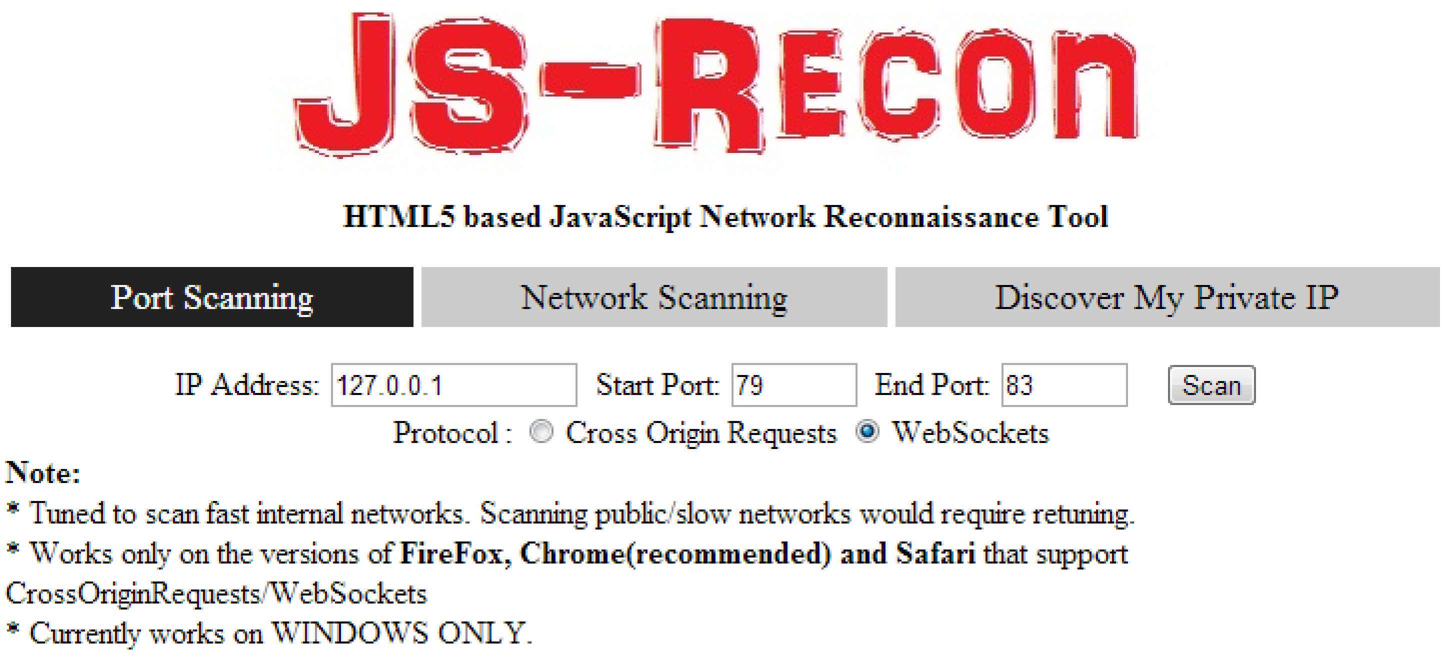
```
GET / HTTP/1.1
Host: domainB.com
Origin: http://domainA.com
…

HTTP/1.1. 200 OK
Content-type: text/html
Access-Control-Allow-Origin: http://domainA.com
…
[data]
```



Cross Origin Resource Sharing

Port / network scanner DEMO:
http://www.andlabs.org/tools/jsrecon.html



HTML5 based JavaScript Network Reconnaissance Tool

IP Address: 127.0.0.1 Start Port: 79 End Port: 83

Protocol : Cross Origin Requests [✓ WebSockets]

Note:
* Tuned to scan fast internal networks. Scanning public/slow networks would require retuning.
* Works only on the versions of **FireFox, Chrome(recommended) and Safari** that support CrossOriginRequests/WebSockets
* Currently works on WINDOWS ONLY.

Iframe sandbox

- Possibility to put Iframe into „sandbox“
- One can specify what is allowed and what not
- `<iframe src=“infected.html” sandbox=“...”>`
- Better control what is allowed than allowScriptAccess attribute in Flash

No sandbox attribute	Javascript runs normally
`sandbox attribute`	JavaScript disabled
`sandbox=“allow-scripts”`	JavaScript enabled ~~document.cookie~~ ~~localStorage()~~ ~~sessionStorage()~~

Content Security Policy

- Protection against XSS attacks
- In HTTP header, server specifies from which domains client can download data

Usage:

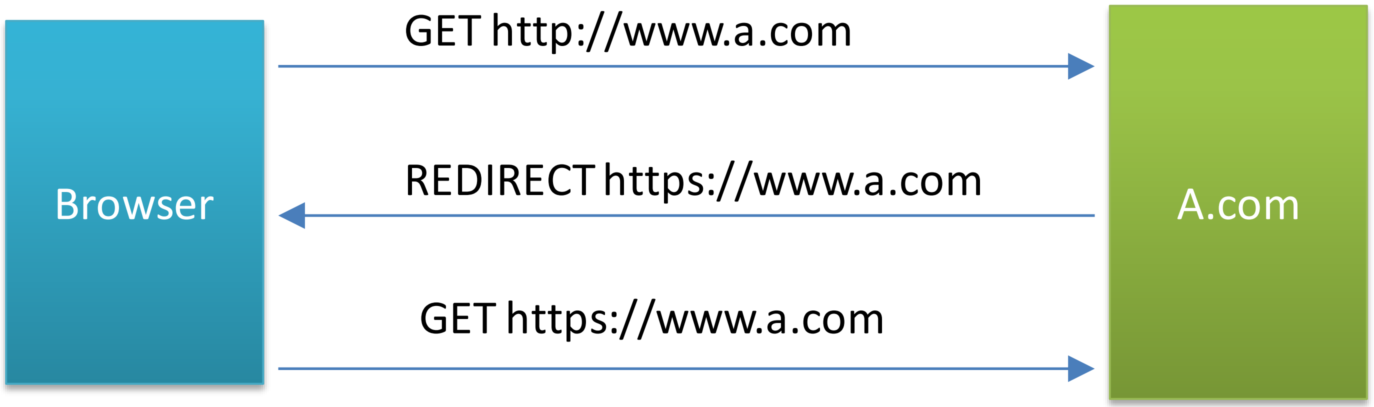
- `Content-Security-Policy: script-src 'self' https://apis.google.com`
- It's possible to specify CSP for different types of data:
  - `connect-src, font-src, frame-src, img-src, media-src, object-src, style-src`

Content Security Policy

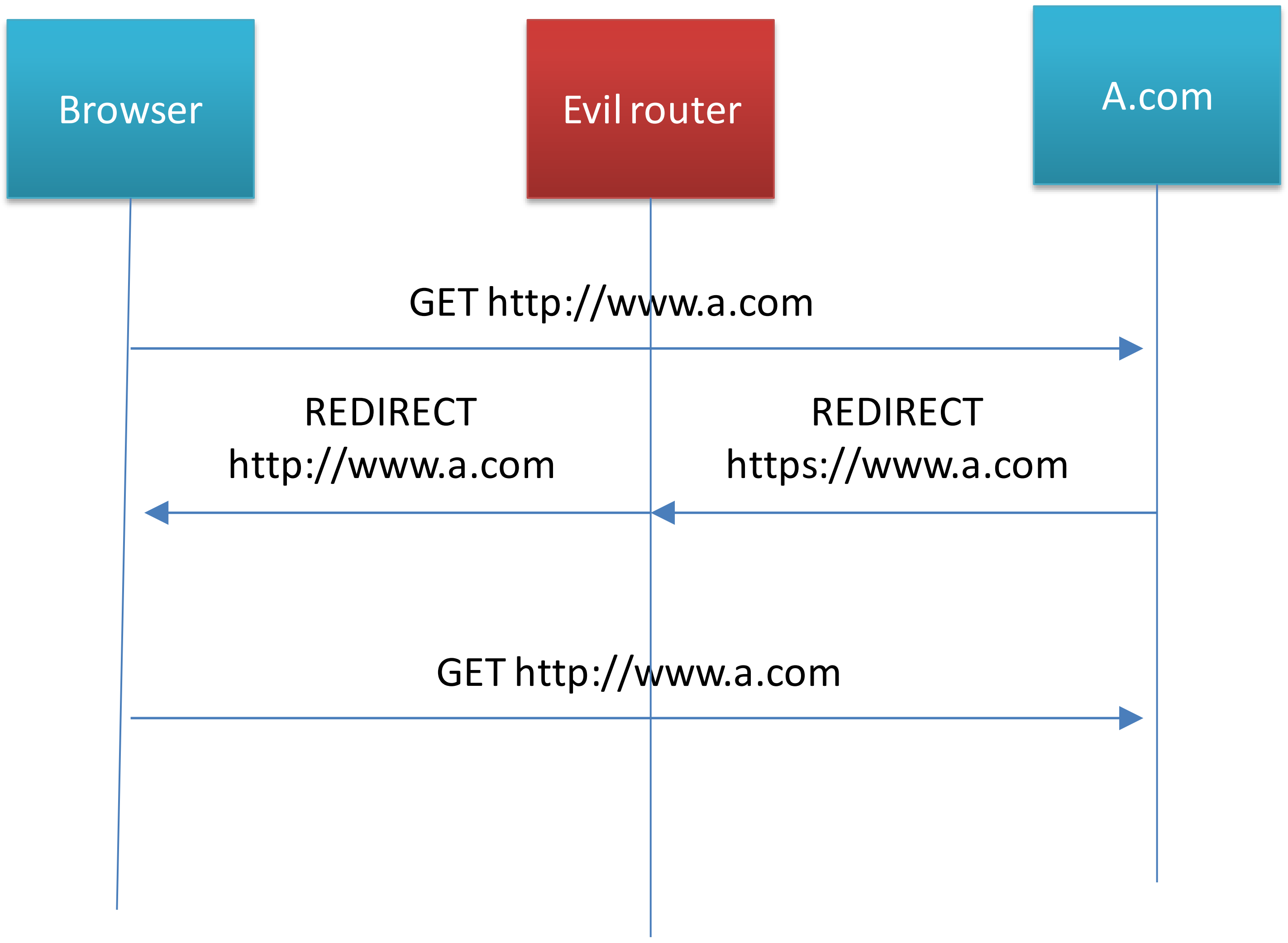
- Need to split resources in separate files
- “inline” <script> is blocked
  - It’s possible to enable inlice scripts via “unsafe-inline” parameter – not recommended (does not prevent XSS attacks)
- Supports error reporting
  - report-uri /my_amazing_csp_report_parser;
  - Browser will send list of blocked resources
  - Only „reporting“ mode supported, no resources are actualy blocked

HTTP Strict Transport Security

- Users visits HTTPS sites mostly via redirect from HTTP site
  - Under some circumstances, attacker can block the redirection
    - SSL stripping attack



HTTP Strict Transport Security
(SSL stripping attack)



HTTP Strict Transport Security

- HTTP header, which minimizes possibility of SSL stripping attack
- `Strict-Transport-Security: max-age=2592000;` *`includeSubDomains`*
  - Browser remembers (for specified amount of time) that it must access the website only via HTTPS
  - All requests to the site are transformed to HTTPS requests
- First request to the server (and first request after HSTS header expires) is vulnerable to SSL stripping attack

X-Frame-Options

- Protection agains „Clickjacking“ attacks
  - attacker opens the desired webpage in Iframe on top of which it places its hidden content.
  - Enables attacker o hijack clicks and key presses
- In HTTP header, it’s possible to restrict viewing page in IFrame:
  - `X-Frame-Options: DENY | SAMEORIGIN | ALLOW-FROM origin`
- Introduced by Microsoft in IE8

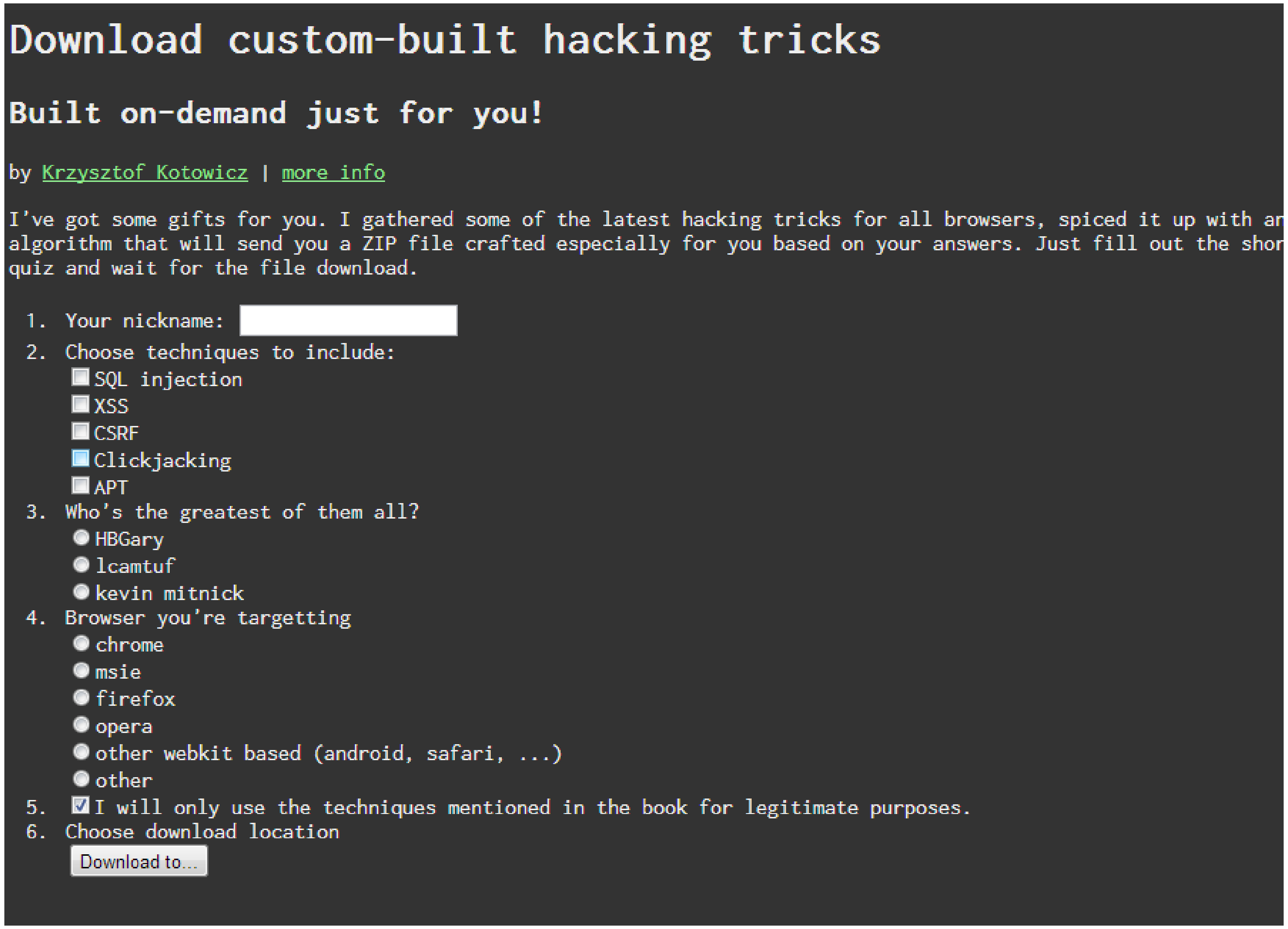
Attack DEMO: file jacking

- It’s possible to upload an entire directory in HTML5
  - <input type=“file” directory>
- http://kotowicz.net/wu/
- Chrome “select folder” dialog can confuse users (see demo)

Attack DEMO: file jacking

- Krystof Kotowicz created a page simulating the file jacking attack
  - Mostly visitor intrested in web security
  - In 13 months 298 clients (217 IP) uploaded some folder (mostly Downloads)
  - Many interesting files:
    - Downloads/#BeNaughtyLive.com/
    - Downloads/#GoLiveTrannies.com/
    - ..
  - Even private data
    - onlinePasswords.txt
    - Faktura_numer_26_2011_<company>.pdf
    - …

Attack DEMO: file jacking



Download custom-built hacking tricks

Built on-demand just for you!

by Krzysztof Kotowicz | more info

I've got some gifts for you. I gathered some of the latest hacking tricks for all browsers, spiced it up with ar algorithm that will send you a ZIP file crafted especially for you based on your answers. Just fill out the shor quiz and wait for the file download.

1. Your nickname:
2. Choose techniques to include:
   - ☐ SQL injection
   - ☐ XSS
   - ☐ CSRF
   - ☐ Clickjacking
   - ☐ APT
3. Who's the greatest of them all?
   - ○ HBGary
   - ○ lcamtuf
   - ○ kevin mitnick
4. Browser you're targetting
   - ○ chrome
   - ○ msie
   - ○ firefox
   - ○ opera
   - ○ other webkit based (android, safari, ...)
   - ○ other
5. ☑ I will only use the techniques mentioned in the book for legitimate purposes.
6. Choose download location

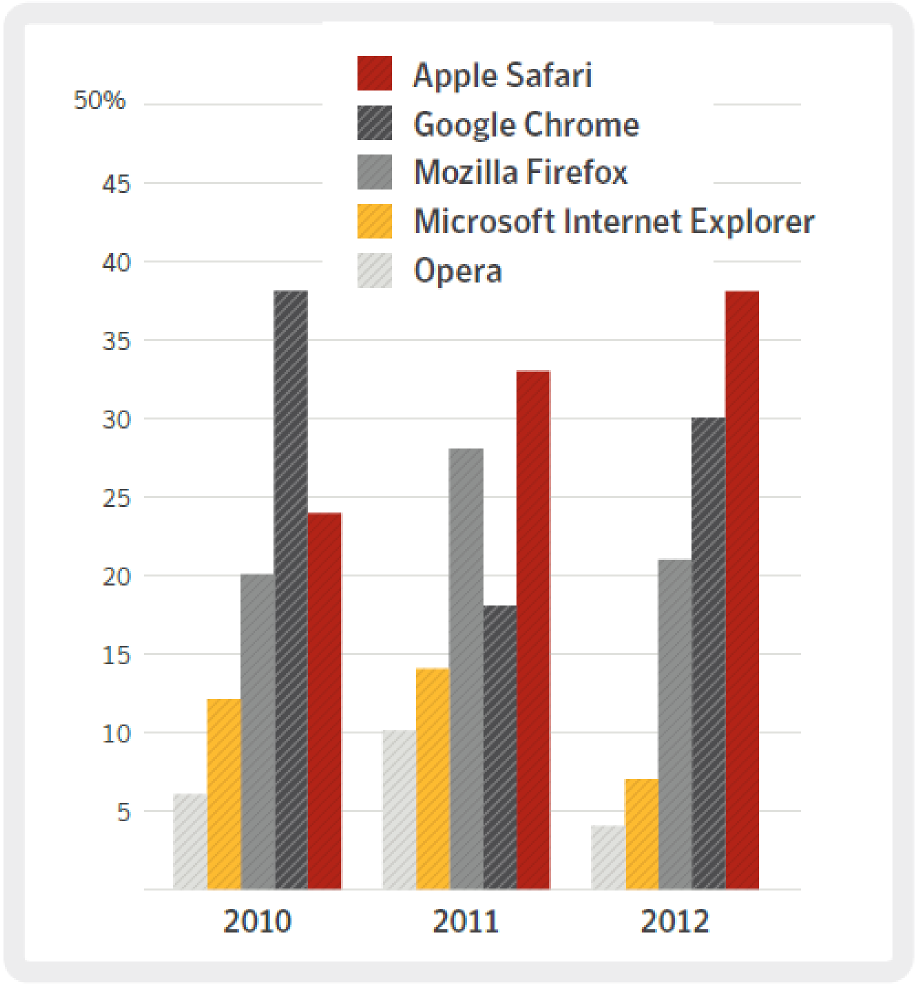
Conclusion

- HTML5 is replacing Flash
- HTML5 has many new existing features and extensions
  - Many features → many vulnerabilities
- Encourage users to update browsers
  - Legacy support is a pain anyway
- When creating web application start with an established JavaScript library (jQuery)
- Adopt HTML5 security features
  - ...to protect users with HTML5-enabled browsers

Symantec Internet Security Threat

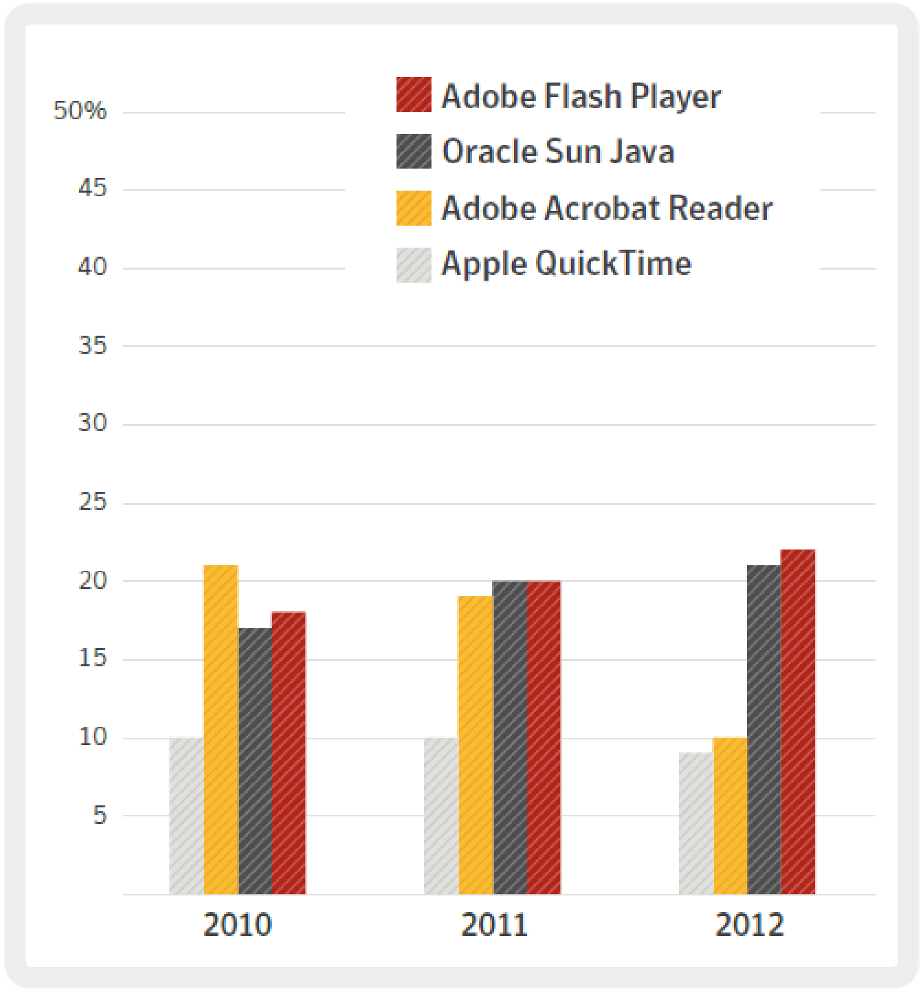
Browser Vulnerabilities 2010 – 2012

Source: Symantec

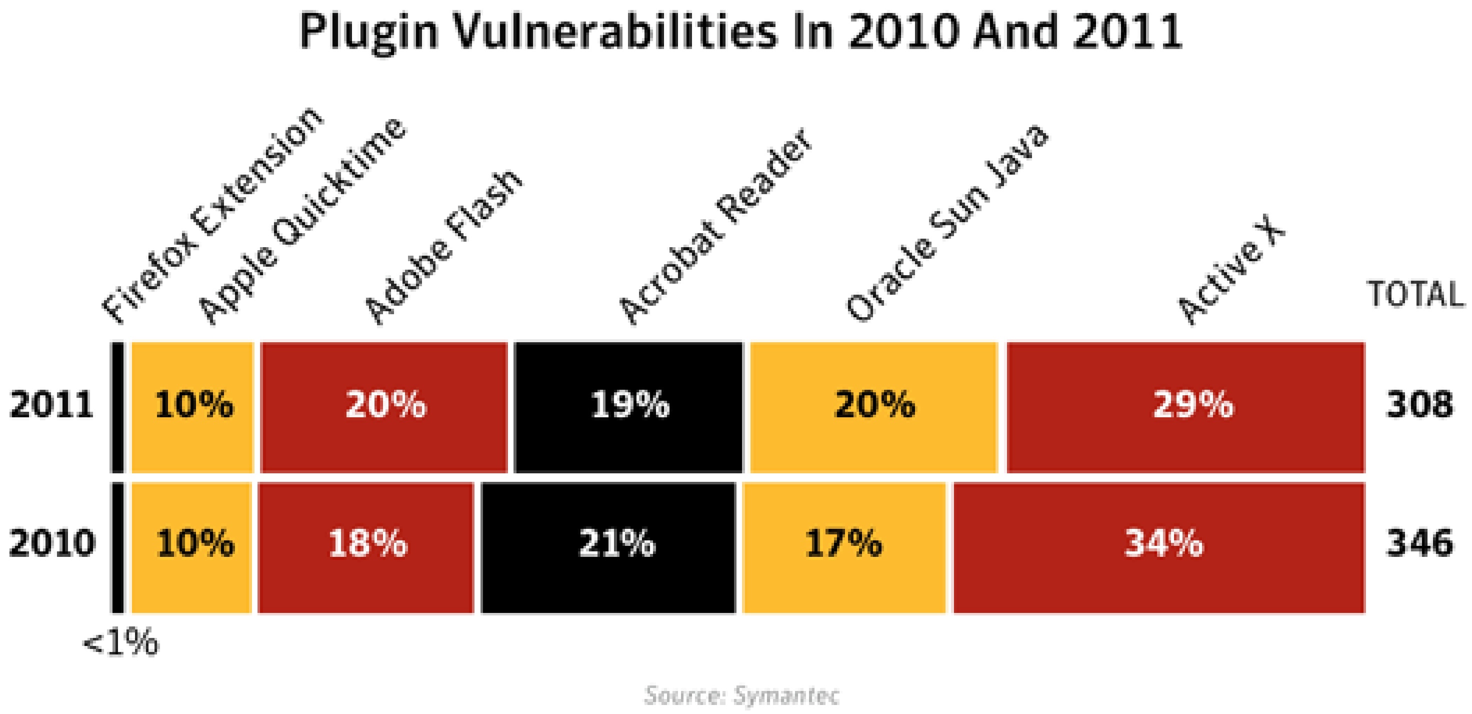


Plug-in Vulnerabilities 2010 – 2012

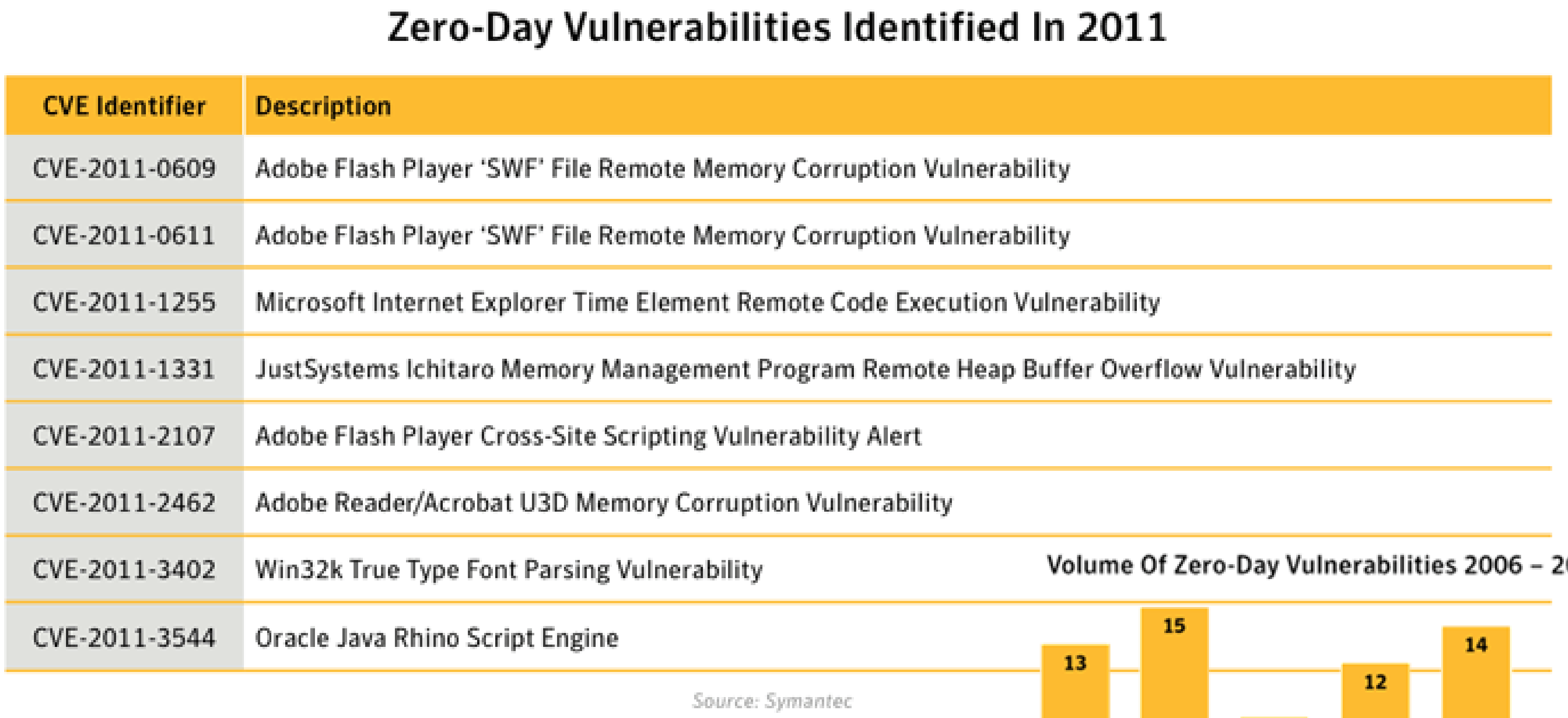
Source: Symantec



Symantec Internet Security Threat Report 2011



Symantec Internet Security Threat Report 2011

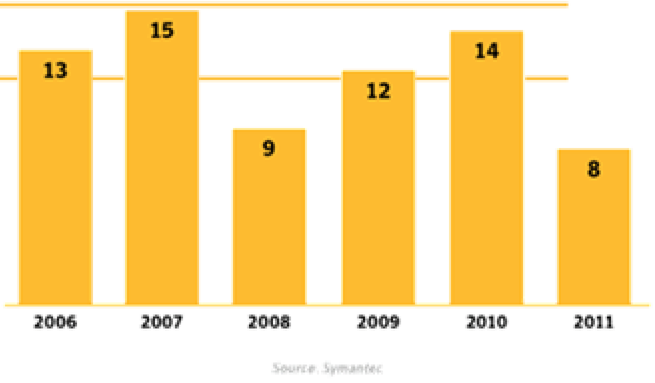


Zero-Day Vulnerabilities Identified In 2011

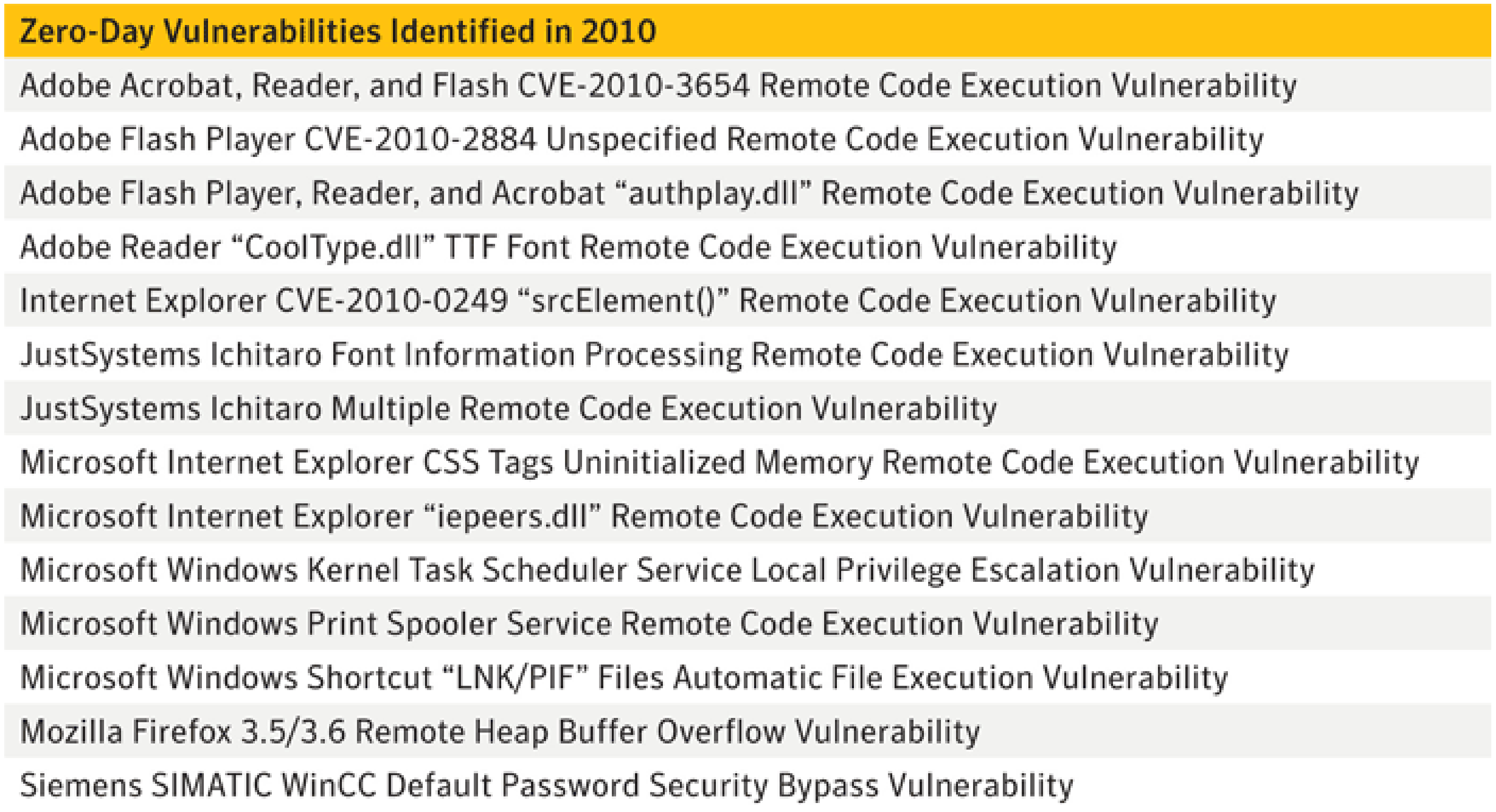
CVE Identifier	Description
CVE-2011-0609	Adobe Flash Player 'SWF' File Remote Memory Corruption Vulnerability
CVE-2011-0611	Adobe Flash Player 'SWF' File Remote Memory Corruption Vulnerability
CVE-2011-1255	Microsoft Internet Explorer Time Element Remote Code Execution Vulnerability
CVE-2011-1331	JustSystems Ichitaro Memory Management Program Remote Heap Buffer Overflow Vulnerability
CVE-2011-2107	Adobe Flash Player Cross-Site Scripting Vulnerability Alert
CVE-2011-2462	Adobe Reader/Acrobat U3D Memory Corruption Vulnerability
CVE-2011-3402	Win32k True Type Font Parsing Vulnerability
CVE-2011-3544	Oracle Java Rhino Script Engine

Source: Symantec

Volume Of Zero-Day Vulnerabilities 2006 – 2011



Symantec Internet Security Threat Report 2010



Zero-Day Vulnerabilities Identified in 2010
Adobe Acrobat, Reader, and Flash CVE-2010-3654 Remote Code Execution Vulnerability
Adobe Flash Player CVE-2010-2884 Unspecified Remote Code Execution Vulnerability
Adobe Flash Player, Reader, and Acrobat “authplay.dll” Remote Code Execution Vulnerability
Adobe Reader “CoolType.dll” TTF Font Remote Code Execution Vulnerability
Internet Explorer CVE-2010-0249 “srcElement()” Remote Code Execution Vulnerability
JustSystems Ichitaro Font Information Processing Remote Code Execution Vulnerability
JustSystems Ichitaro Multiple Remote Code Execution Vulnerability
Microsoft Internet Explorer CSS Tags Uninitialized Memory Remote Code Execution Vulnerability
Microsoft Internet Explorer “iepeers.dll” Remote Code Execution Vulnerability
Microsoft Windows Kernel Task Scheduler Service Local Privilege Escalation Vulnerability
Microsoft Windows Print Spooler Service Remote Code Execution Vulnerability
Microsoft Windows Shortcut “LNK/PIF” Files Automatic File Execution Vulnerability
Mozilla Firefox 3.5/3.6 Remote Heap Buffer Overflow Vulnerability
Siemens SIMATIC WinCC Default Password Security Bypass Vulnerability

References

- Flashsec Wiki
- OWASP – Finding vulnerabilities in flash applications
- Adobe Flash Player 10 security white paper
  - http://www.adobe.com/devnet/flashplayer/articles/flash_player10_security_wp.html
- Symantec Internet Security Threat Report
  - http://www.symantec.com/threatreport/
- HTML 5 rocks
  - http://www.html5rocks.com/
- http://blog.kotowicz.net
- http://www.andlabs.org