Overview

• The general problem
• Plug-ins
• ActiveX
  • Authenticode
• Java
  • SecurityManager
• JavaScript
• XUL

The Basic Problem

• Increasingly, personal computers consist of both traditionally installed (from disk) and web installed software
• The web installed software includes
  • Applications – netbeans, googleEarth, etc.
  • Utilities – acrobat reader
  • Plug-ins for browsers – Flash
  • ActiveX controls
  • Java applets
  • JavaScript rich web pages
• These programs do not have the same assurance of issuer that would accompany traditional programs
Risks of Downloaded Code

- Downloaded code can:
  - Spend money – via mechanisms such as phone calls or other hooks in vulnerable software (wire transfers)
  - Release private data stored on your machine or processed by it
    - Packet sniffers for credit card numbers
    - Audio/video devices
  - By-pass firewalls by encrypting data and hiding it in email or other normal packets

History of Exploits

- CardShark was designed to look like a conventional screensaver, but instead it monitored keyboard activity for credit card numbers – it was intended to promote a system for protected web payments
- David.exe was a download to look for pornography that used modems to make calls to numbers that billed by the minute
- Chaos-Quicken was a demonstration program that could make wire transfers via Quicken
- ILoveYou was a worm that exploited vulnerabilities in Outlook to infect millions of machines

Helpers and Plug-ins

- Until browsers were built to allow plug-ins, they had to run in separate windows and were called Helper applications
- Plug-ins
  - Microsoft, at the same time (1996), introduced the notion of ActiveX controls for the same functionality
  - Historically, plug-ins required a manual download and installation – increasingly, it is done automatically
  - The basis of the problems with this code is that they are binary executables that can do anything
Plug-ins versus Applets

- Netscape allowed websites to download applets which were (theoretically) constrained so as to run in a restricted sandbox.
- Plug-ins, which were manually loaded had a greater range of allowable actions:
  - Full access to data on the machine
  - Network access
- Once installed, plug-ins are automatically used for the downloads they serve
- They can also take other actions about which they may or may not inform users
- Further, there is no guarantee that even a trusted benign plug-in does not have a vulnerability that will be later exploited

Security an Plug-ins

- To protect the user, methods were developed to allows developers to "sign" code
- Browsers have various controls to provide security alerts related to plug-ins
- Evaluation Plug-in security
  - Users should only download plug-ins from sites they trust
  - Check the web for vulnerabilities related to the plug-in (www.tenablesecurity.com)
  - Carefully check the attached certificates

Microsoft ActiveX

- ActiveX components, "ocx's" are used for many purposes, but are frequently downloaded when <object> tags are encountered in web pages
- ActiveX controls are basically unrestricted in what they can do on a windows machine
- ActiveX controls are treated differently – more trusted – than plug-ins
- ActiveX controls will not run by Internet Explorer unless they are digitally signed with a "software publisher’s certificate"
- Controls to be run in web browsers must be marked “safe for scripting”
- Existing OCX's and COM objects can be inspected using the oleviewer that is part of the windows resource kit
ActiveX Downloads

- When the browser encounters an Object tag in a webpage:
  - The browser checks to see if the plug-in is already downloaded (ClassID)
  - If not, it downloads it, checking the digital signature according to the security level
  - It then sets the parameters specified in the object
  - It then runs the plug-in/ActiveX control

Sample Object Tag

```html
<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F490D93"
    width="200" height="200" align="baseline"
    codebase="http://javaweb.eng/plugin/jre-1_3-win.exe#Version=1,3,0,0">
    <PARAM NAME="code" VALUE="XYZApp.class">
    <PARAM NAME="codebase" VALUE="html/">
    <PARAM NAME="type" VALUE="application/x-java-applet;version=1.3">
    <PARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
    <COMMENT>
        <EMBED type="application/x-java-applet;version=1.3" width="200" height="200" 
            align="baseline" code="XYZApp.class" codebase="html/" 
            pluginspage="http://javaweb.eng/plugin/plugin-install.html">
            <NOEMBED></COMMENT>
    <NOEMBED></OBJECT>
```
**Authenticode**

- Whereas Sun created a “safe” sandbox in which to run Java Applets, Microsoft chose to allow full access to its code, but to provide assurance of authorship via a digital signature.
- The signature is a digitally signed checksum for the objects plus a public key certificate – a “software publisher’s certificate” – signed by a certification authority.
- It can be used to sign:
  - Windows exe’s
  - ActiveX controls
  - Java bytecode
- Windows 2000 and IE both implement Authenticode checks

**Questions about Authenticode**

- Even though an organization signs code, it doesn’t mean an employee didn’t put malicious code in it.
- Even if there is no ”malicious code” it doesn’t mean the code can’t be exploited.
- A legitimate certificate might have been awarded to a malicious organization.
- There is no audit trail provided for signed code that allows us to know who did what, or even that the signing information was not altered by the code.

**Java as Safe Code**

- Java was originally developed as the green project as a way of running small programs on virtually any platform via the Java Virtual Machine (JVM).
- Most early Java for the desktop (Applets) failed for lack of a quality GUI.
- Java did establish a significant footprint as a server side technology via servlets and JSP.
- Java code, run in web browsers as byte code applets was considered to be safer than running binaries.
Java as a Safe Language

• Java was written to be a more secure language
  • Garbage collection versus manual memory management
  • Built in bounds checking to avoid buffer overflows
  • Elimination of pointers
  • Strong typing
  • A mandated exception handling system

Java Security

• As originally written, Java lacked many things – one of which was security
  • Java had a weak GUI – given that it was intended originally for device control
  • Java had weak security – given that it was originally imagined as used by a closed community – device programmers
  • Java has been greatly expanded, providing for a full GUI API, enhanced security model, and many relevant security related packages

Java’s Security Features

• Java has increased its sensitivity to security in several ways:
  • The JVM operates in a sandbox that is a restricted virtual space
  • The Java SecurityManager class is called by any operation that might be deemed dangerous
    • Generally local apps have full privileges and web apps have minimal privileges.
  • The SecurityManager class is protected from malicious manipulation by the Class Loader which examines classes as they are loaded
  • Further, a bytecode Verifier insures that the bytecodes were produced by a java compiler and contain no malicious code
Security Policies

• Java security policies have become very complex, but as initially envisioned, web browsers would use one of three:
  • Don’t run Java Programs
  • Run Java programs with differential privileges based on the source
  • Allow full privileges to Java programs
• Browsers provide users with a number of options regards how these policies are implemented
  • IE uses security zones
  • Firefox uses an all or none policy

Web Browser Security for Java

• In general, the JVM in web browsers is restricted from the following operations:
  • Can’t read, write, rename, or delete local files
  • Can’t initiate a network connection other than to the computer from which the download occurred
  • Can’t open a listening socket
  • Can’t create a ClassLoader or SecurityManager
  • Can’t run system programs
  • Can’t create windows other than browser windows

IE’s Security Zones

• IE simplifies the process of defining which sites to trust by defining “zones”.
  • Intranets can be identified by the use of Microsoft’s networking protocols or unqualified domain names
  • My computer includes everything on my local machines
  • The internet is everything else, but is related to:
    • Trusted zones – which have been marked as trusted
    • Restricted zone – which have been marked as untrusted
• Security for each of the zones is set as one might expect, but it can be tailored in a variety of ways
Java Security Issues

• For the most part, it appears that SUN has fixed most of the major implementation and design flaws related to security that were reported in the late 1990’s
• This doesn’t mean that there are not new issues, but most of the security discussions petered out around 2002

JavaScript

• While the security issues related to Java have been reduced since 2000, the issues related to JavaScript seem to be on the increase
  • JavaScript functionality has increased dramatically
  • Network connections
  • File I/O
  • The use of cross site scripting has increased
• JavaScript also has a number of issues related to cross browser compatibility

JavaScript Flaws

• JavaScript can create forms that contain browser and user information that can be automatically submitted to third part sites
  • Email can be sent in your name
  • URL history lists can be obtained
  • Files can be uploaded to a hostile site
• Malicious JavaScript can be constructed to execute denial of service attacks
• A JavaScript can freeze a browser, or crash an OS
  • By opening windows of various sorts
  • while (1) {alert(“gotcha”);}
  • N.B. windows versus Unix behavior
  • By executing recursive code
More Issues

- JavaScript can change the browser appearance, open a variety of windows, manipulate the anchor references.
- JavaScript with cute programming can further confuse the matter.
- JavaScript can be transferred to a Web Server as innocent text that is subsequently passed from the server to another browser via another page, being executed on the second machine.
- AJAX can be used with server proxies to relay information to other locations.

XUL

- Matters get yet more complex with Firefox and XUL which allow the very fabric of the browser to be manipulated and overlayed.
- When JavaScript is used with XUL and XPI it is possible to read and write files – this is not network delivered JavaScript, but more like a plug-in.
- MORE TO SAY HERE LATER.