Browser Security

John Mitchell

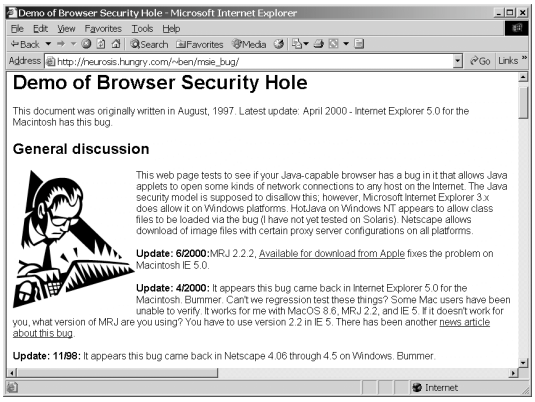
Question from last time: Purify

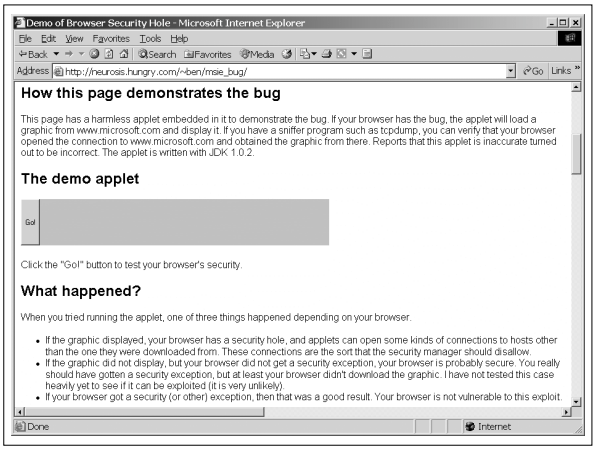
- Goal
  - Instrument a program to detect run-time memory errors (out-of-bounds, use-before-init) and memory leaks
- Technique
  - Works on relocatable object code
    - Link to modified malloc that provides tracking tables
  - Memory access errors: insert instruction sequence before each load and store instruction
  - Memory leaks: GC algorithm

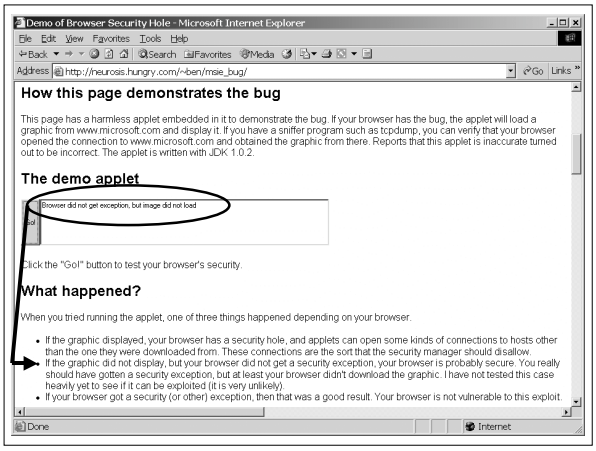
Browser security

- Browser uses network and local disk
  - Potential for outside access to local data
- Brower interprets code from network
  - HTML, JavaScript, ActiveX, Java
- Browser installs, executes plug-ins
  - Acrobat, Shockwave, …
- Malicious code can pose risks
  - Consume resources
  - Steal information
  - Compromise system

A browser is an operating system







INTERNETWEEK.com CONNECTING THE ENTERPRISE, CUSTOMERS & SUPPLIERS Tuesday, February 12, 2002

Microsoft Issues New IE Browser Security Patch

By Richard Karpinski

- Microsoft has released a security patch that closes some major holes in its Internet Explorer browser
- The so-called "cumulative patch" fixes six different IE problems ...
- Affected browsers include Internet Explorer 5.01, 5.5 and 6.0.
- Microsoft rated the potential security breaches as "critical."

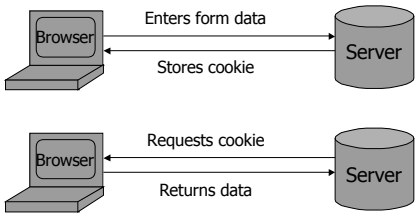
Latest patch addresses:

- A buffer overrun associated with an HTML directive ... Hackers could use this breach to run malicious code on a user's system.
- A scripting vulnerability that would let an attacker read files on a user's systems.
- A vulnerability related to the display of file names ... Hackers could … misrepresent the name of a file ... and trick a user into downloading an unsafe file.
- A vulnerability that would allow a Web page to improperly invoke an application installed on a user's system to open a file on a Web site.
- … more …

Tour of security issues

- Cookies
- JavaScript
- ActiveX
- Java
  - Most of lecture devoted to Java
    - Representative case, more developed security model
- Using a network proxy to increase security
- Plug-ins ?

Cookies



- Http is stateless protocol; cookies add state
  - Other method: modify URL

Cookie issues

- Policy
  - Cookie from site S can be returned to site S *only*
- Problems
  - Cookies maintain record of your browsing habits
  - Sites can share this information (e.g., doubleclick)
  - Attacks could invade your "privacy"

    08 Nov 2001

    Users of Microsoft's browser and e-mail programs could be vulnerable to having their browser cookies stolen or modified due to a new security bug in Internet Explorer (IE), the company warned today.

JavaScript

- Language executed by browser
- Used in many attacks
  - Cookie attack from last slide:

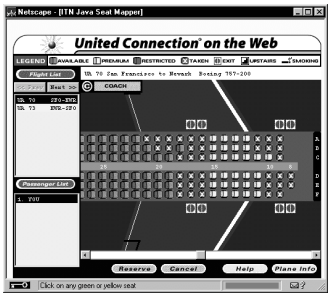
    With the assistance of some JavaScript code, an attacker could construct a Web page or HTML-based e-mail that could access any cookie in the browser's memory or those stored on disk ...

ActiveX

- ActiveX controls reside on client's machine, activated by HTML object tag on the page
  - ActiveX controls are not interpreted by browser
  - Compiled binaries executed by client OS
  - Can be downloaded and installed
- Security model relies on three components
  - Digital signatures to verify source of binary
  - IE policy can reject controls from network zones
  - Controls marked by author as *safe for initialization*, *safe for scripting* which affects the way control used

Once accepted, installed and started, no control over execution

Java Applet



- Local window
- Download
  - Seat map
  - Airline data
- Local data
  - User profile
  - Credit card
- Transmission
  - Select seat
  - Encrypted msg

Security Risks

- Annoyance or inconvenience
  - Display large window that ignores mouse input
  - Play irritating sound and do not stop
  - Consume CPU cycles, memory, network bandwidth …
- Export confidential information
  - Communication is generally possible
  - Prevent access to password file, credit card number, …
  - Subtle attack: trick dialog boxes ...
- Modify or compromise system
  - Delete files, call system functions

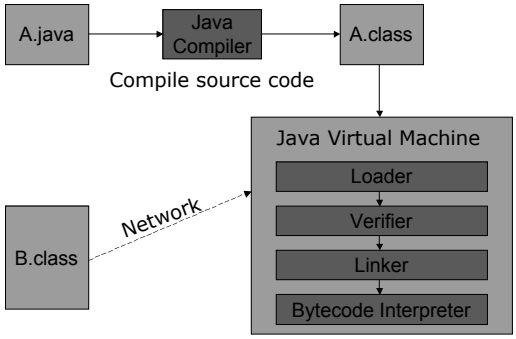
Mobile code security mechanisms

- Examine code before executing
  - Java bytecode verifier performs critical tests
- Interpret code and trap risky operations
  - Java bytecode interpreter does run-time tests
  - Security manager applies local access policy
- Beyond the Browser: code modification
  - Replace standard calls by calls to "safe" versions
  - Check parameters to standard methods to make sure they are in appropriate ranges

Java Background

- Compiler and Virtual Machine
  - Compiler produces bytecode
  - Virtual machine loads classes on demand, verifies bytecode properties, interprets bytecode
- Why this design?
  - Portability
    - Transmit bytecode across network
  - Minimize machine-dependent part of implementation
    - Do optimization on bytecode when possible
    - Keep bytecode interpreter simple

Java Virtual Machine Architecture



Class loader

- Runtime system loads classes as needed
  - When class is referenced, loader searches for file of compiled bytecode instructions
- Default loading mechanism can be replaced
  - Define alternate ClassLoader object
    - Extend the abstract ClassLoader class and implementation
  - Can obtain bytecodes from network
    - VM restricts applet communication to site that supplied applet

Verifier

- Bytecode may not come from standard compiler
  - Evil hacker may write dangerous bytecode
- Verifier checks correctness of bytecode
  - Every instruction must have a valid operation code
  - Every branch instruction must branch to the start of some other instruction, not middle of instruction
  - Every method must have a structurally correct signature
  - Every instruction obeys the Java type discipline

Last condition is fairly complicated .

Bytecode interpreter

- Standard virtual machine interprets instructions
  - Perform run-time checks such as array bounds
  - Possible to compile bytecode class file to native code
- Java programs can call native methods
  - Typically functions written in C

Type Safety of JVM

- Load-time type checking
- Run-time type checking
  - All casts are checked to make sure type safe
  - All array references are checked to be within bounds
  - References are tested to be not null before dereferenc
- Additional features
  - Automatic garbage collection
  - NO pointer arithmetic

    If program accesses memory, the memory is allocated to the program and declared with correct type

Why is typing a security feature?

- Java sandbox mechanisms rely on type safety
- Example
  - Unchecked cast lets applet make any system call

```
int (*fp)()     /* variable "fp" is a function pointer          */
...
fp = addr;      /* assign address stored in an integer var  */
(*fp)(n);       /* call the function at this address            */
```

How do we know verifier is correct?

- Many early attacks based on verifier errors
- Formal studies prove correctness
  - Abadi and Stata
  - Freund and Mitchell
    - Found error in initialize-before-use analysis

JVM uses stack machine

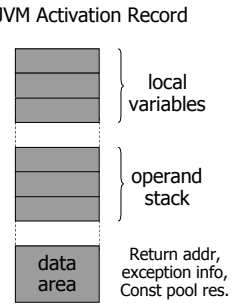
- Java

```
Class A extends Object {
    int i
    void f(int val) { i = val + 1;}
}
```

- Bytecode

```
Method void f(int)
    aload 0   ; object ref this
    iload 1   ; int val
    iconst 1
    iadd      ; add val +1
    putfield #4 <Field int i>
    return
```

refers to const pool



Java Object Initialization

```
Point p = new Point(3);
p.print();

1:  new Point
2:  dup
3:  iconst 3
4:  invokespecial <method Point(int)>
5:  invokevirtual <method print()>
```

- No easy pattern to match.
- Multiple refs to same uninitialized object.

Bug in Sun's JDK 1.1.4

- Example:

variables 1 and 2 contain references to two different objects

```
1:  jsr 10
2:  store 1
3:  jsr 10
4:  store 2
5:  load 2
6:  init P
7:  load 1
8:  use P
9:  halt
10: store 0
11: new P
12: ret 0
```

General Security Risks

- Denial of Service
  - Tie up your CPU, network connection, subnet, …
- Steal private information
  - User name, email address, password, credit card, …
- Compromise your system
  - Erase files, introduce virus, ...

Java Security Mechanisms

- Sandboxing
  - Run program in restricted environment
    - Analogy: child's sandbox with only safe toys
  - This term refers to
    - Features of loader, verifier, interpreter that restrict program
    - Java Security Manager, a special object that acts as access control "gatekeeper"
- Code signing
  - Use cryptography to determine who wrote class file
    - Info used by security manager

Java Sandbox

- Four complementary mechanisms
  - Class loader
    - Separate namespaces for separate class loaders
    - Associates *protection domain* with each class
  - Verifier and JVM run-time tests
    - NO unchecked casts or other type errors, NO array overflow
    - Preserves private, protected visibility levels
  - Security Manager
    - Called by library functions to decide if request is allowed
    - Uses protection domain associated with code, user policy
    - Enforcement uses stack inspection

Security Manager

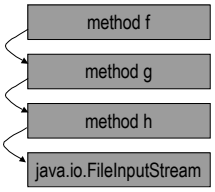
- Java library functions call security manager
- Security manager object answers at run time
  - Decide if calling code is allowed to do operation
  - Examine protection domain of calling class
    - Signer: organization that signed code before loading
    - Location: URL where the Java classes came from
  - Uses the system policy to decide access permission

Sample SecurityManager methods

checkExec	Checks if the system commands can be executed.
checkRead	Checks if a file can be read from.
checkWrite	Checks if a file can be written to.
checkListen	Checks if a certain network port can be listened to for connections.
checkConnect	Checks if a network connection can be created.
checkCreate ClassLoader	Check to prevent the installation of additional ClassLoaders.

Stack Inspection

- Permission depends on
  - Permission of calling method
  - Permission of all methods above it on stack
    - Up to method that is trusted and asserts this trust



Many details omitted

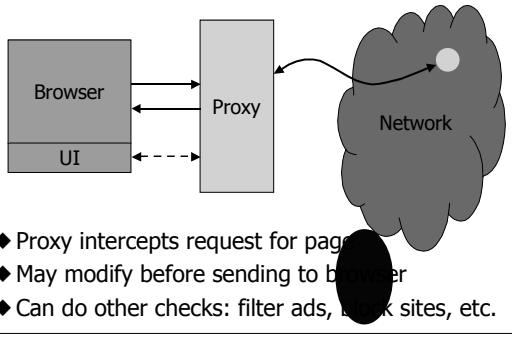
Stories: Netscape font / passwd bug; Shockwave plug-in

Beyond JVM security

- JVM does not prevent
  - Denial of service attacks
    - Applet creates large windows and ignores mouse
  - Certain network behavior
    - Applet can connect to port 25 on client machine, forge email (on some implementations)
  - URL spoofing
    - Applet can write false URL on browser status line
  - Annoying behavior
    - Applet can play loud sound
    - Applet can reload pages in new windows

Additional Security

Modify code in proxy [Shin, M…]



- Proxy intercepts request for pag
- May modify before sending to b er
- Can do other checks: filter ads, k sites, etc.

Bytecode Modification Techniques

- Class-level replacement
  - Define subclass of library (or other) class
  - Replace references to class with subclass (const pool)
  - Works because of subtyping
  - Not possible if class is final
- Method-level replacement
  - Change function calls to new function
  - Generally, check or modify arguments and call original function

Sample bytecode modification

- SafeWindow class
  - Subclass of standard Window class
    - Do not allow windows larger than maximum
    - Do not allow more than max number of windows
- Restrict network activity
  - Replace call to Socket object constructor
    - Do not allow socket connection to port 25
- Maintain appearance of browser window
  - Replace calls to AppletContext methods
    - Displayed URL must match actual hyperlink

Browser security

- Many issues
  - Browser sits between network and local disk
  - Interpret commands from untrusted sites
  - Manage execution of native code on client machine

We'll see many of these issues in other forms when we discuss OS security, network security