Java Security: From HotJava to Netscape and Beyond.

Drew Dean Edward W. Felten Dan S. Wallach
Department of Computer Science
Princeton University

Presented by Yuhong Cai

Introduction

- This article introduces many significant flaws found in HotJava, Netscape and JDK.
- Issues about java type-safe, independent name space and byte-code verifier are worth pages of explanation.
- The scenarios mentioned in the part of taxonomy of Java Bugs are not straightforward(it can't be, otherwise the attackers will be too happy, but it's also hard for us to understand them).
- I recommend one excellent on-line book www.securingjava.com which gives more details about java security.
- I also recommend www.digicrime.com and www.rstcorp.com/hostile-applets/ where we can have unforgettable experience.

Presentation Contents

- Java Semantics(The important concepts in java security)
- Three examples out of taxonomy of Java Bugs
  - Denial of Service (malicious applet which consumes all your resources)
  - Killing competent applets (malicious applet which kills other applets but themselves)
  - You're Not My Type (Attack applet which can bypass the Java SecurityManager and attack your system)
- Java Language and Bytecode Weaknesses
- Conclusions

Java Semantics

(Some corner stones of java security architecture)

- **Java should always be type-safe language.**
- **Any code is trusted if it is loaded from the local filesystem**
- **Applets cannot directly make system calls**
  - SecurityManager module approves dangerous operations
- **Applets are forbidden to**
  - access the file system
  - open sockets, except back to their home
  - interfere with other applets
  - learn about the local environment

Example(1)
(Denial of Service)

- **Denial of Service (the next-best goal of crackers)**

  Attacks that prevent someone from using his or her machine are called denial-of-service attacks in the security community.
- **Various guises of denial of service:**
  - Consuming all available CPU cycles.
  - Allocating every last bit of memory.
  - Hogging all possible screen space.
- **Netscape 3.0 is a denial-of-service victim of the applet on the next slide.**

Example (1)(cont'd)

```
Public class MaliciousApplet extends Applet implements Runnable {
                public void init() {
                   // Determine how many seconds the thread should sleep before kicking in;
                 }
                public void start(){
                  // Create and start the offending thread in the standard way;
                }
                public void stop(){
                  // Redefine the stop() method to null for the thread.;
                }
                public void run(){
                  // Start calculating in an infinite loop or any other naughty things;
                }
```

Reference:http://www.rstcorp.com/hostile-applets/Consume.java

Example(2)

(killing off the Competition)

- **Killing Off the Competition**

  An Assassin applet first spawns a monitoring thread to watch for applets from another site then kills the threads of any incoming applets.

- **Main idea of this malicious applet**
  - Starting with the current thread group, ascend to the root thread group.
  - From the root, recursively descend through all threads and thread groups below.
  - Kill each thread encountered (but not self)
  - Navigator 3.0 can be the victim of this applet.

  Reference:http://www.rstcorp.com/hostile-applets/AppletKiller.java

Example(3)

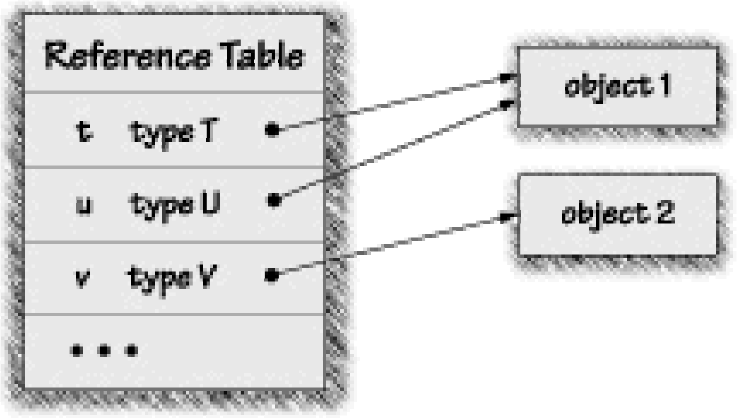
(You're not my type)

- **You are Not My Type (A type-confusion attack, very dangerous)**
  - An applet creates two pointers to the same object-with incompatible type tags.
  - The Java system is in trouble.
  - The applet can write into that memory address through one pointer, and read it through another pointer.
  - Finally the applet can bypass the typing rules of Java, completely undermining its security.
  - Navigator 3.0 beta 5.0 has this bug, but it's fixed in Navigator 3.0 beta 6.0.

  To be continued

Example(3)(cont'd)

If the following situation occurs in a java system, this type-confusion will results heavy system attack.(There is a good case to describe this problem in www.securingjava.com/chapter-five/chapter-five-7.html)



Java Language and Bytecode Weakness

- **Public variables are dangerous**
  - Why are they writable across name spaces?
- **Java's package mechanism**
  - Not as useful as parameterized module system(e.g. Standard ML's functors)
  - Hierarchical module system allows hierarchical protection.
- **Abstract Syntax Trees vs. Bytecode**
- **ASTs easier to type check**
  - No need for global dataflow analysis
- **Asts have same semantics as language**
  - Bytecode has its own semantics
- **Comparable compilation speed**

Conclusion

- **Remote code is inevitable for the Web.**
- **Java is promising, but has important bugs and design issues.**
- **Like java applets which can be naughty on client side, the misbehavior of java servlets can results more serious problems on server side.**
- **Strong security measures can allow more functionality for untrusted applets without compromising privacy and integrity (Java 2 security package).**

- ***Acknowledgement:*** *I would like to thank professor Clark Thomborson, he's given me many helpful suggestions to make this presentation clear and concise.*

Questions

Interesting questions:

- *Have you ever suffered from malicious attack from java applets?*
- *Would you like to disable java or javascript in your browser for security aim?*

Tough question:

- ***** Could you tell any difference between iexplorer security mechanism and netscape security mechanism?*