

From Branon Martindale

- The presidents of the National Academies said yesterday that the Bush administration was going too far in limiting publication of some scientific research out of concern that it could aid terrorists.
- Specifically, they said, the administration's policy of restricting the publication of federally financed research it deemed "sensitive but unclassified" threatened to "stifle scientific creativity and to weaken national security."
- The category of "**sensitive but unclassified**" was poorly defined, the presidents said in a "Statement on Science and Security in an Age of Terrorism."
- "Experience shows that vague criteria of this kind generate deep uncertainties among both scientists and officials responsible for enforcing regulations," the statement said.
- Indeed, the policy, experts said, had already resulted in the administration's withdrawing of thousands of reports and papers from the public domain.



Sensitive But Unclassified

- 9/11
- **Combat Operations**
 - Afghanistan
 - Philippines
 - Balkans
 - Future Operations in Southwest Asia
- **Continued Computer Security Issues**
- **SBU**
 - Unclassified
 - No criminal sanctions for disclosure
 - exception: information covered by Privacy Act, other legislation or court order



Cookies

- **Where web servers store information about their customers**
 - searching large customer databases on server costly
- **HTTP requests do NOT automatically identify individual users**
 - Thus easier to use a cooperating browsers' customer side
 - Server requests browser to store a cookie that contains information the server will use the next time the client calls
 - **.netscape/cookies**
- **Cookies give browsers the chance to create stateful HTTP sessions**
- **Privacy**
 - cookies stored by the browser create client profiles



Language Design Decisions (Java)

- The language itself should make it more difficult for programs to create damage.
- The execution environment provides mechanisms for access control
- The security policies enforced by the execution environment have to be set correctly



Java Review: Applets vs. Applications

From: *Java in a Nutshell – Flanagan*

- “A **program** in Java consists of one or more class definitions, each of which has been **compiled** into its own **.class file of Java Virtual Machine object code.**”
 - One of these classes must define a method `main()`, which is where the program starts running.
 - To invoke a Java program you run the Java interpreter, *java*, and specify the name of the class that contains the `main()` method.
- A Java applet is NOT an application – it is a Java class that is loaded and run by an already running Java application such as a web browser or an applet viewer.
- Note: Ada 95 has this capability – i.e. “Adapplets.”



Security for Executable Java Applets (Objectives)

- Applets do not get access to the user's file system
- Applets cannot obtain information about the user's name, email address, machine configuration, etc.
- Applets may make outward connections only back to the server they came from
- Applets can only pop-up windows that are marked "untrusted"
- Applets cannot reconfigure the system, e.g. by creating a new class loader or a new security manager



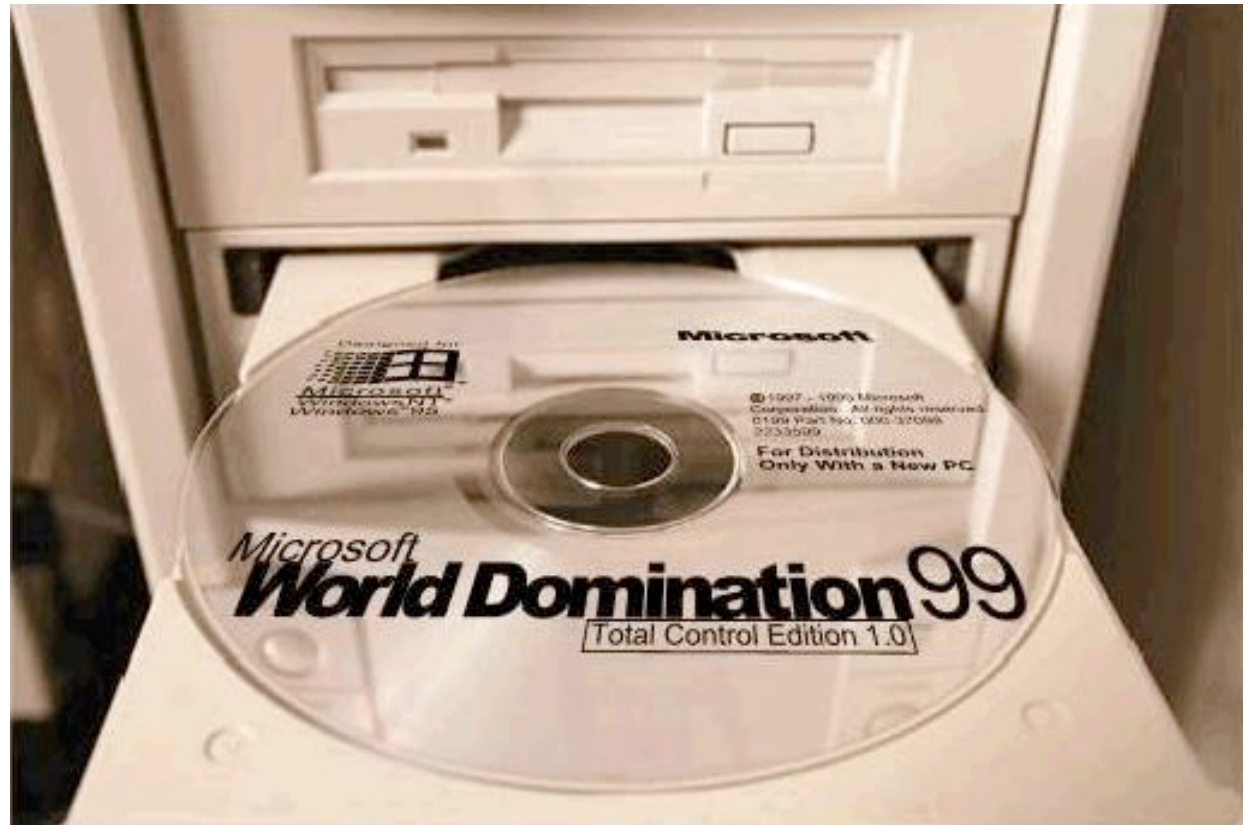
Applications versus Applets

- An applet loaded across the network it is not allowed to:
 - read/write/delete files on the client file system
 - no use of `File.delete()` method or `sys` calls `rm` or `del`
 - rename files on the client file system
 - no use of `File.renameTo()` or `mv` or `rename` commands
 - conduct directory operations
 - content listing
 - check for existence of a file
 - obtain file information – size, type and modification time stamp
 - conduct network operations
 - create a network connection to any computer other than the host from which it originated
 - listen for or accept network connections on any port in the client system
 - specify any network control functions – `SocketImplFactory`, etc.....
 - read or define any system properties
 - run or exit any program
 - no use of `Runtime.exec()`, `System.exit()` or `Runtime.exit()` methods
 - load dlls on the client system using `load()` or `loadLibrary()`
 - thread creation or manipulation
 - create a new `ClassLoader` or `SecurityManager`
 - define classes that are part of packages on the client system

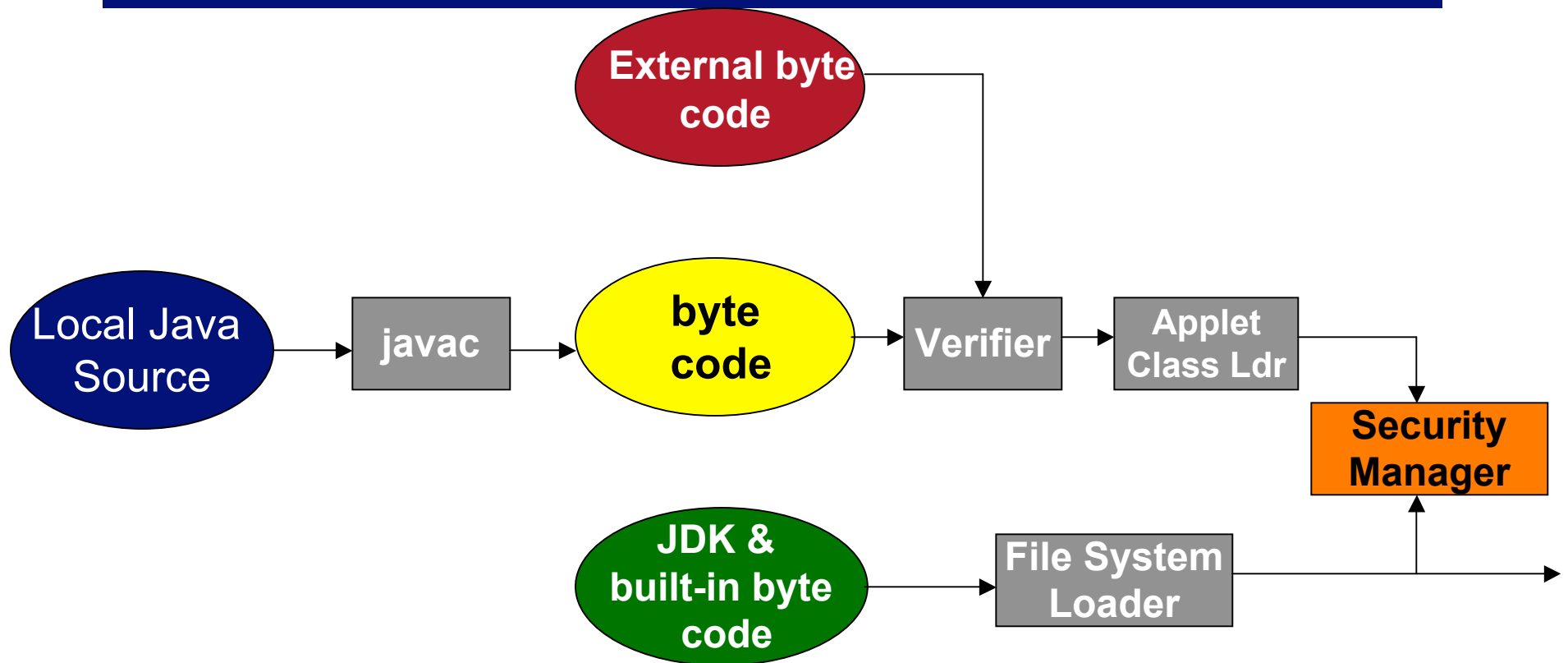


Environment for Applets

- Users cannot rely on prior acquaintance and trust relationship with the source of an applet
- Few users are willing to rule personally on each access request made by an applet
- Client's operating system cannot be expected to offer any protection



Three roads for Java byte code

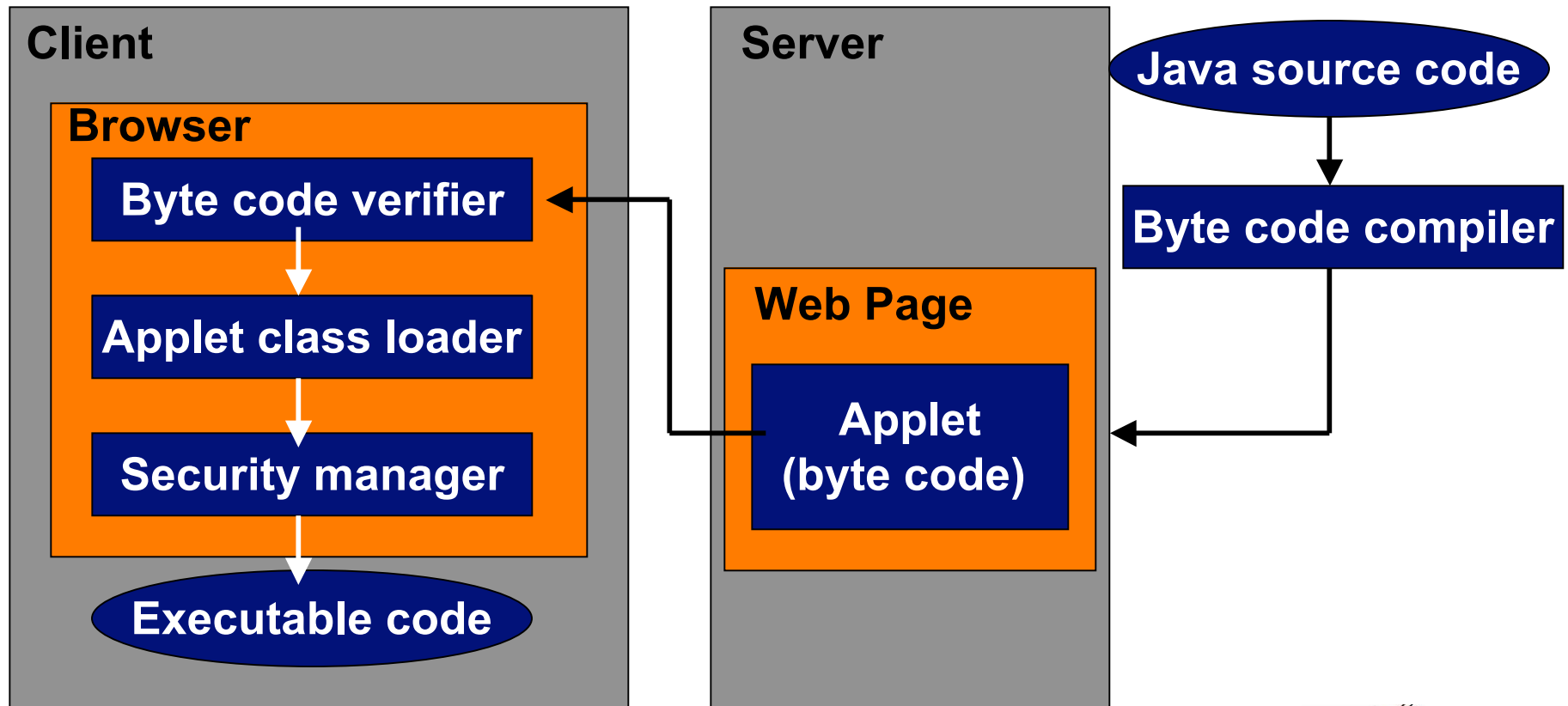


- **External byte code** loaded across the network must be verified
- Locally developed **byte code** is subject to the same checks unless it is part of the CLASSPATH
- **Byte code from the JDK distribution (and other classes in CLASSPATH)** does NOT pass through the verifier, **may** be checked by Security Manager



Java Sandbox

- Executable Content (applets) from remote web sites

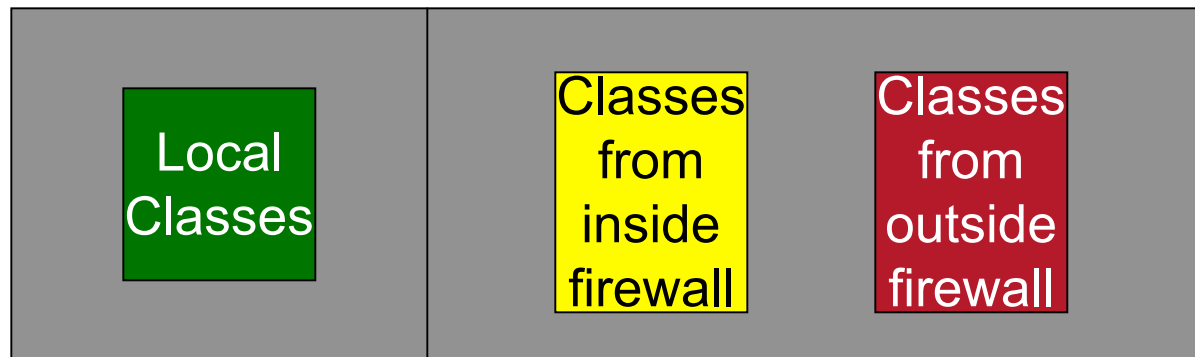


Byte Code Verifier

- **Checks for:**
 - the class file is in the proper format
 - stacks will not overflow
 - all operands have the correct type
 - there will be no data conversion between types
 - all references to other classes are legal
- **Byte code verifier reduces the workload on the interpreter**
 - guaranteed code properties do not have to be checked again
- *However, security still depends on the run-time environment*



Class Loader in a Dynamic Environment



- The Java environment has classes arriving and departing dynamically
- The class loader divides classes that it loads into several distinct name spaces according to where the classes came from
- Local classes are kept distinct from classes loaded from other machines
- Furthermore, these outside classes are protected from each other.



Applet Class Loader

- **Class loader protects the integrity of the run-time environment**
- **Applets must not be allowed to create their own class loaders**
 - Applets are handled by the applet class loader
- **Java comes with its own class library**
 - The CLASSPATH environment variable specifies the location of built-in classes
 - The security issues associated with altering CLASSPATH should be obvious
- **“Spoofing” of the CLASSPATH can be avoided by:**
 - If the applet class loader first searches the built-in classes in the local name space
 - Then expand search to the class making the request



Security Manager

- **Reference Monitor of the Java Security Model**
 - Performs run-time checks on ‘dangerous’ methods
- **Java classes are grouped into packages**
 - packages facilitate rudimentary access control to classes
- **Variables and methods can be declared as follows:**
 - **Private:** only the class creating the variable or method has access
 - **Protected:** only the class creating the variable or method and its subclasses have access
 - **Public:** all classes have access
 - **None of the above:** only classes within the same package have access



Security Manager Functions

- **Prevent installation of new Class Loaders.**
 - The Class Loader's job is to keep the name spaces properly organized.
 - Because things like file I/O permission will depend on whether or not a class is local, the Class Loader has an important job.
 - **Must not be subject to spoofing**
- **Protecting threads and thread groups**
 - Not fully functional....
- **Controlling the creation of OS programs**
- **Controlling access to OS processes**
- **Controlling file system operations such as read & write**
 - access to local files strictly controlled
- **Controlling socket operations such as connect and accept**
- **Controlling access to Java packages (or groups of classes)**



Some Compromises of the “Sandbox”

- **MSIE Cache Exploit**
 - <http://www.alcrypto.co.uk/java/>
- **Advanced Type Systems in Computing**
 - <http://www.cs.nps.navy.mil/research/languages/>
- **Mark LaDue’s “Public Enemy”**
 - <http://www.cigitallabs.com/resources/hostile-applets/>
- **Others I chose not to experiment with:**
 - `diskhog.java`
 - `triplethreat.java`
 - `mutator.java`



Microsoft Internet Explorer - "Where do you want (your data) to go today?"

- The object of the exercise here is to open a connection to a port on the local machine, and provide a two-way pipe back to a remote machine on the Internet.
 - This is achieved by using the Java `net.socket` class to talk to the local machine, and the `showDocument()` thingy for the remote.
 - This exploit relies on the fact that Java behaves differently when loaded across the net, to a load from local hard disk.
 - When loaded across the net, the applet is not allowed to open a network socket to anything other than the server that delivered it in the first place
 - (see <http://www.javasoft.com/sfaq/#socket> for details).
 - This is enforced by the centralized security manager class. However, if the applet is loaded from local disk, this limitation is relaxed, allowing a socket to be opened on the browsing machine.



Type Systems for Secure Remote Evaluation

- The project aims to improve our understanding of the role of type systems in programming languages.
 - Type systems provide a very elegant separation of concerns.
 - Static analyses are typically much easier to reason about when captured by a logical framework such as a type system.
 - Implementations of the analyses are separate algorithmic issues that have their own soundness and completeness proof obligations.
 - This project is concerned with developing new type systems and techniques for formal proofs of semantic soundness, algorithmic issues, and computational lower bounds for these systems.
- This effort aims to identify the rudiments of a provably-secure programming language.
 - It requires formulation of appropriate security and safety properties so that one can prove with respect to a formal semantics that every well-typed program cannot violate these properties.
 - For example, it would be nice to prove that every well-typed Java Applet when executed by Netscape does not cause Netscape to crash. Clearly, there isn't such a proof as evidenced by enabling Java in them and [clicking here](#) to run a tiny (killerApp)let.



PublicEnemy.java by Mark LaDue

This Java application directly attacks Java class files. Given a target directory, it searches that directory and all subdirectories for Java class files. Once a class file is located, PublicEnemy alters the contents of its "access_flags" for the class, its fields, and its methods. The results are the following:

1. The class becomes public.
2. Any "static" or "volatile" fields remain as such; "final" fields become "non-final"; "transient" fields become "non-transient;" and "private" or "protected" fields become "public," while "public" fields remain so.
3. Any "abstract, "native," "synchronized," or "static" methods remain as such; "final" methods become "non-final;" and "private" or "protected" methods become "public," while "public" methods remain so.

This should open the class to the maximum amount of inspection and abuse without directly affecting its ability to run. Note that the size of the resulting class is the same as the original. The ability to modify Java class files on the fly is just the skill that a Java Platform Virus will require. The fact that it's this easy bodes ill....



Summation of Web Threats

	Threats	Consequences	Countermeasures
Integrity	<ul style="list-style-type: none"> •Modification of user data •Trojan horse browser •Modification of memory •Modification of message traffic in transit 	<ul style="list-style-type: none"> •Loss of information •Compromise of machine •Vulnerability to all other threats 	<ul style="list-style-type: none"> •Cryptographic checksums
Confidentiality	<ul style="list-style-type: none"> •Eavesdropping on the Net •Info theft from server •Info theft from client •Info about network configuration •Info about which clients talk to server 	<ul style="list-style-type: none"> •Loss of Information •Loss of Privacy 	<ul style="list-style-type: none"> •Encryption, •Web Proxy
Denial of Service	<ul style="list-style-type: none"> •Killing of user threads •Flooding machine with bogus threats •Filling up disk or memory •Isolating machines by DNS attack 	<ul style="list-style-type: none"> •Disruptive •Annoying •Prevent user from getting work done 	<ul style="list-style-type: none"> •Difficult to prevent
Authentication	<ul style="list-style-type: none"> •Impersonation of legitimate users •Data Forgery 	<ul style="list-style-type: none"> •Misrepresentation of user •Belief that false information is valid 	<ul style="list-style-type: none"> •Cryptographic techniques



Static versus Dynamic Type Checking

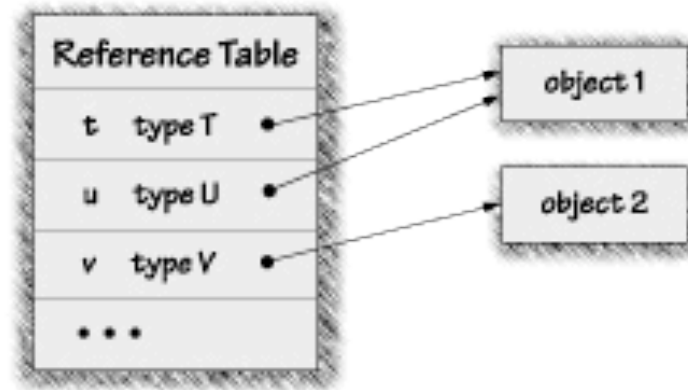
- **Java has no dynamic memory allocation**
 - does not allow you to cast object or array references into integers or vice-versa
 - does not allow “pointer arithmetic”
 - does not allow you to compute the size in bytes of any primitive type or object
- **Dynamic type checking is inefficient**
 - to improve performance, Java uses static type checking
 - faster, but less secure than say, Ada
- **In a type-confusion attack, a malicious applet creates two pointers to the same object-with incompatible type tags.**
 - When this happens, the Java system is in trouble.
 - The applet can write into that memory address through one pointer, and read it through another pointer.
 - The result is that the applet can bypass the typing rules of Java, completely undermining its security.



Type Confusion Attack Example

- The applet has two pointers to the same memory: one pointer tagged with type T and one tagged with type U. Suppose that T and U are defined like this:

```
class T {  
    SecurityManager x;  
}  
class U {  
    MyObject x;  
}
```

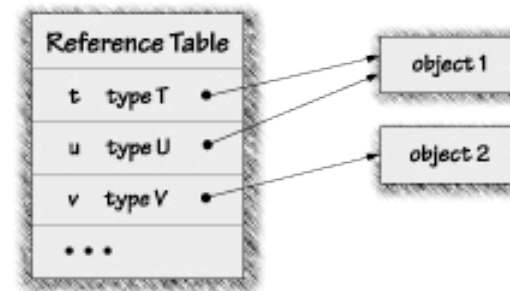


- Now the applet can run code like this:
T t = the pointer tagged T;
U u = the pointer tagged U;
t.x = System.getSecurity(); // the Security Manager
MyObject m = u.x;



Exploit Results

- The result is that the object ends up with a pointer, tagged as having type `MyObject`, to the memory representing Java's Security Manager object.
- By changing the fields of `m`, the applet can then change the Security Manager, even though the Security Manager's fields have been declared private.
- While this example showed how type confusion can be used to corrupt the Security Manager, the tactic may be exploited to corrupt virtually any part of the running Java system.




<http://www.securingjava.com/chapter-five/chapter-five-7.html>



Conclusion

- **Fundamental engineering tradeoff:**
 - **Functionality versus Security**
 - **Always an inverse relationship**
 - **Evident in many aspects of security**



You don't know her. But she might know you. The fact is, hackers are virtually everywhere. And if your network isn't protected by a BorderWare firewall, it's not as secure as it should be.

BorderWare is the first firewall designed to be as smart about time and money as it is about security. It's the only firewall that's up and running in hours, not days.

It offers total transparency to all authorized users, so existing software and procedures don't need to be modified, and no one needs retraining. It runs its own diagnostics.

What's more, it doesn't require expensive workstations. In fact, BorderWare operates on an Intel® processor.

As well, it has everything you need to link to the Internet: Mail, News, WWW, FTP and DNS. And it combines packet filtering with both application-level and circuit-level gateways.

BorderWare also enables you to define proxies for secure and specialized applications through the firewall. It responds to attacks and initiates alarms. And is configured to grow as you and your enterprise do.

It's an impressive list of features. But it has to be. Because the whole world, unfortunately, is watching.

<http://www.borderware.com>

BORDERWARE
Nobody Comes Close

All trademarks are the property of their respective owners.

Call us at 1-800-334-8195 or +1(416) 368-7157. Fax +1(416) 368-7789. 20 Toronto Street, Suite 400, Toronto, Ontario, Canada M5C 2B8.

