Java in(security)

Adam Gowdiak
Security Explorations

INTRODUCTION
About the speaker

- Nearly 15 years of Java hacking experience
  - 2002 Bytecode Verifier flaws, JIT compiler flaw
    - first exploit turning memory corruption into type confusion
  - 2004 First public J2ME hack
  - 2005 Java SE
    - discovery and reporting of Reflection API flaws and RMI attack vector
  - 2008 J2ME security vulnerabilities
  - 2011 Java based SAT TV boxes
    - first malware for set-top-boxes
  - 2012 Java SE (Method Handles API bugs)
  - 2013 Oracle Java Cloud Service
  - 2014 Oracle Database Java VM
  - 2015 Google App Engine Java security sandbox

INTRODUCTION
About Security Explorations

- Security and vulnerability research company from Poland
- Came to life in a result of a true passion of its founder for breaking security of things and analyzing software for security defects
- Our ambition is to conduct quality, unbiased, vendor-free and independent security and vulnerability research
- One of our missions is to increase general awareness of users and vendors in the area of computer and Internet security
  - Pro Bono security research

INTRODUCTION
Presentation Goal

- Analysis of key problems related to Java platform security, its ecosystem and vendors
- Discussion of the issues that contributed to the failure of an original Java security promise and its safe mobile code execution paradigm in particular
- Presentation of real life examples of the traps some vendors fell victim of when trying to address Java security problems

JAVA SECURITY
Why important ?

- An attractive target for attackers
  - Java runs on 1.1 billion desktops
  - 930 million Java Runtime Environment downloads each year
- Powerful attack vector
  - multi platform (Windows, Linux, Solaris, MacOS, AIX, …),
  - reliable exploitation (non memory corruption issues)
- Within interest of nation states
  - US, Israel, China, Russia, …

„We are interested in JVM vulnerability research […] We want to acquire several unknown high severity (remote code execution) vulnerabilities…"

JAVA SECURITY

Original design principles

- *„Java must enable the development of secure, high performance, and highly robust applications on multiple platforms in heterogeneous, distributed networks."*
- *„Java is designed to operate in distributed environments, which means that security is of paramount importance."*

The Java Language Environment,
James Gosling, Henry McGilton

JAVA SECURITY
Original design principles (2)

- Designed with a security in mind
  - The paradigm of safe execution environment for untrusted, mobile code
- Access control at classes, methods and fields level
  - private, protected, public, default (package)
- Memory safety
  - Strict type checking
    - Type safety
  - Garbage collection
    - No memory pointers
    - No free() operation
  - Immutable, safe strings representation
  - Runtime checks for arrays

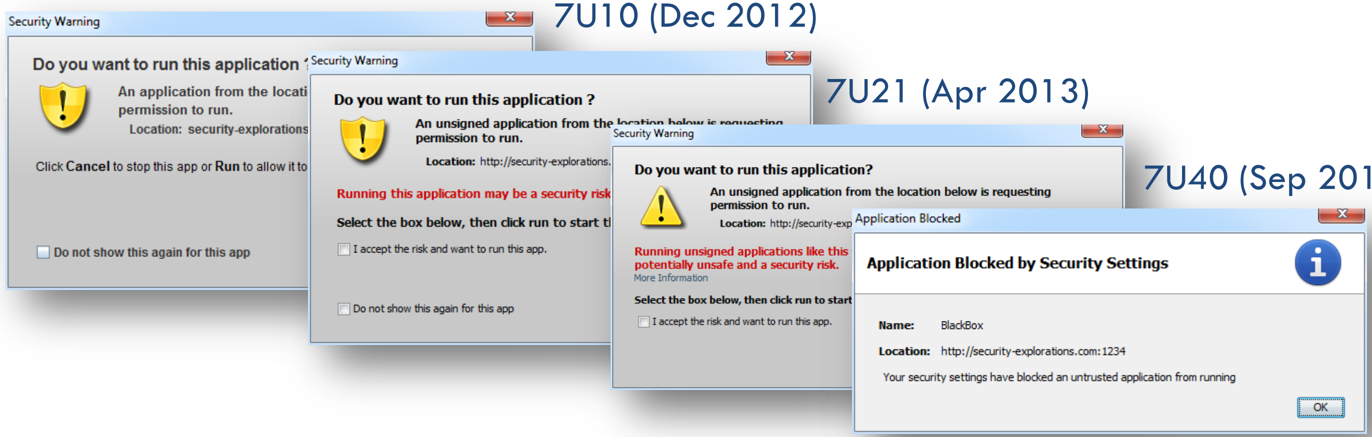
JAVA SECURITY
Nearly 20 years later

- Responsible for a vast number of attacks in recent years
  - Java represented 91 percent of all Indicators of Compromise in 2013 (The Cisco 2014 Annual Security Report)
  - Used by cybercriminals and nation state actors against companies from technology, pharmaceutical, commodities and legal sectors
    - Primary exploit vector in attacks against Apple, Facebook, Microsoft and Twitter (2013)
- Java considered a huge security risks to users and businesses
  - Triggering action by US government bodies
    - Department of Homeland Security (warning to disable Java)
    - Federal Trade Commission (investigation over deceptive Java security updates)
      - FTC contacted Security Explorations in Jun 2013

JAVA SECURITY

Nearly 20 years later (2)

- Key feature of the language denounced by the vendor
  - Unsigned code execution deemed a security risk
  - Security dialog prompt before Java Applet / Java Web Start application launch



7U10 (Dec 2012)

Security Warning

Do you want to run this application

An application from the locati
permission to run.

Location: security-explorations

Click Cancel to stop this app or Run to allow it to

Do not show this again for this app

7U21 (Apr 2013)

Security Warning

Do you want to run this application ?

An unsigned application from the location below is requesting
permission to run.

Location: http://security-explorations.

Running this application may be a security risk

Select the box below, then click run to start t

I accept the risk and want to run this app.

Do not show this again for this app

7U40 (Sep 2013)

Security Warning

Do you want to run this application?

An unsigned application from the location below is requesting
permission to run.

Location: http://security-exp

Running unsigned applications like this
potentially unsafe and a security risk.
More Information

Select the box below, then click run to start

I accept the risk and want to run this app.

7U51 (Jan 2014)

Application Blocked

Application Blocked by Security Settings

Name: BlackBox

Location: http://security-explorations.com:1234

Your security settings have blocked an untrusted application from running

JAVA SECURITY
How did we get here ?

- Technical and non-technical issues that contributed to the failure of an original Java security promise and its safe mobile code execution paradigm in particular
- A combination of problems related to Java platform security, its ecosystem and vendors

PLATFORM WEAKNESSES

Complexity

- Oracle „Secure Coding Guidelines for Java SE" reflects the complexity of Java security model
- The guidelines might be too difficult to understand even for skilled developers
- CERT from Carnegie Mellon University compared C and Java secure coding guidelines
  - *„Java has 168 coding rules compared to just 116 for C"*
  - *„If you are writing Java code to manage unprivileged Java code (such as an applet container), you are subject to about as many severe rules as if you are writing C code"*

PLATFORM WEAKNESSES
Complexity outcome

- One needs to be very careful about the implementation of privileged code sequences
  - Malicious arguments to privileged operations
  - Inheritance / methods overloading
    - Bypassing security checks (time of check / time of use issues)
  - Reflection API
    - Diverting execution to arbitrary methods
  - Serialization
    - Bypassing security checks in constructors
  - Class Loading
    - Linking with restricted classes
  - …

PLATFORM WEAKNESSES
Complexity outcome (2)

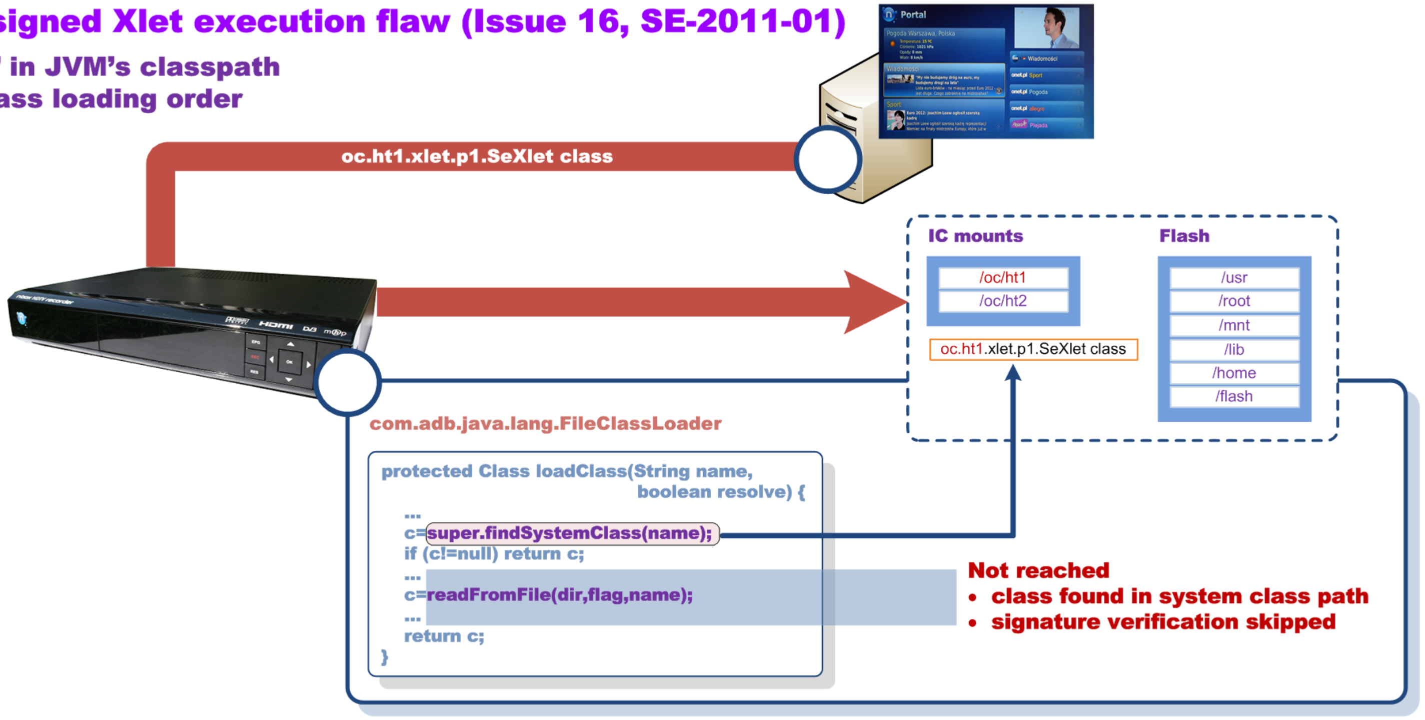
- Small, innocent looking Java security issues do matter in Java
  - they can be often combined together to gain full privileges
- Even little modifications of security sensitive components can ruin security
  - Class Loaders / Security Manager (ADB)
  - Class Loaders / Reflection API (Google)
- The need of a deep knowledge and understanding of
  - Java security model and JVM operation in general
  - past / current Java attack techniques

PLATFORM WEAKNESSES

Complexity (ADB sample)

Unsigned Xlet execution flaw (Issue 16, SE-2011-01)

- „/” in JVM’s classpath
- class loading order



oc.ht1.xlet.p1.SeXlet class
Portal
IC mounts
/oc/ht1
/oc/ht2
oc.ht1.xlet.p1.SeXlet class
Flash
/usr
/root
/mnt
/lib
/home
/flash
com.adb.java.lang.FileClassLoader

```
protected Class loadClass(String name,
                          boolean resolve) {
    ...
    c=super.findSystemClass(name);
    if (c!=null) return c;
    ...
    c=readFromFile(dir,flag,name);
    ...
    return c;
}
```

Not reached
- class found in system class path
- signature verification skipped

PLATFORM WEAKNESSES

Reflection API / method handles API

- Reflection API and method handles API do not fit Java security model well
  - security relying on a caller class / lookup class
  - Reflection API in wide use by system classes
  - dynamic access to classes, fields, methods and constructors
- Security got even more complex!
  - the need to guard access to objects denoting restricted classes or their members
  - skipping / ignoring Reflection API frames in security checks
  - adding extra stack frames prior to certain method invocations
    - sensitive callers / bound method handlers
  - invocation trampolines (`MethodUtil.invoke`)

PLATFORM WEAKNESSES
`AllPermission` equivalent privileges

- Java permissions that alone could be elevated to `AllPermission`
  - `createClassLoader`
  - `accessClassInPackage.sun`
  - `setSecurityManager`
- Java security bypassing privileges
  - `accessDeclaredMembers`
  - `suppressAccessChecks`
- Some are way too often granted to user code
  - Oracle Java Cloud Service (SE-2013-01)
  - Google App Engine for Java (SE-2014-02)
    - Permission to create Class Loaders

PLATFORM WEAKNESSES
`sun.misc.Unsafe`

- The „official backdoor" class with a functionality to break Java memory safety (and security)
  - Memory read and write primitives (`putInt`, `getInt`, ...)
  - Native `defineClass()` method that allows to inject arbitrary, fully privileged classes into a system class loader namespace
- It is not available by default to unprivileged code
- It was exploited in dozens of Java security sandbox escapes over the recent decade (since 2005)
  - Source of vulnerabilities / privilege elevation vector

PLATFORM WEAKNESSES
sun.misc.Unsafe (2)

- Sun Microsystems warned that `sun.*` packages are not part of the supported, public interface

  „Why Developers Should Not Write Programs That Call 'sun' Packages"

  http://www.oracle.com/technetwork/java/faq-sun-packages-142232.html
- Many vendors use it in their code (i.e. Google App Engine)
- The problem is about the availability of the unsafe functionality
  - Instrumental in breaking memory safety / proceeding with attacks against host platforms
- The class should be gone years ago!
  - Jigsaw irrelevant in this context

PLATFORM WEAKNESSES

Little exploitation countermeasures

- Sandbox escape can be directly leveraged to break type and/or memory safety
  - `sun.misc.Unsafe` functionality
  - specially crafted Reflection API setup
    - SAT TV set-top-boxes (SE-2011-01)
    - Oracle Java Cloud Service (SE-2013-01)
- Native code execution could be easily gained
  - Certain JIT memory regions mapped as RWX
    - memory area pointed by methodOop's adapter handle
  - Artifacts facilitating interaction with the OS (native signal handler)
    - Oracle Java Cloud Service (SE-2013-01)
    - Google App Engine (SE-2014-02)

PLATFORM WEAKNESSES

Native layer

- Native code is prone to all types of vulnerabilities known from „unsafe” languages such as C / C++
- Java 2D was in particular vulnerable to all sorts of security issues
  - Image formats / font parsing code / raw graphics primitives (rasters)
- Pure native code vulnerabilities partially mitigated by security countermeasures / sandboxing at OS level
- Native vulnerabilities can be however successfully exploited by turning them into type confusion flaws (pure Java level flaws)
  - Netscape JIT Compiler flaw (LSD, 2002)
  - Apple Quicktime for Java (Issue 22, SE-2012-01)

VENDORS RELATED ISSUES

Questionable SW security assurance processes

"Whenever Oracle makes an acquisition, acquired product lines are required to conform to Oracle policies and procedures, including those comprising Oracle Software Security Assurance„

Oracle VP, May 2013

VENDORS RELATED ISSUES

Questionable SW security assurance processes (2)

- Dozens of Reflection API based security vulnerabilities in Java SE 7 code
  - The vulnerability class known to the vendor
  - 20+ issues reported to Sun Microsystems in 2005
- Information about associated risks included in *Secure Coding Guidelines for Java SE* since 2010
  - *Guideline 6-3 Safely invoke standard APIs that bypass SecurityManager checks depending on the immediate caller's class loader*
- 1.5 years to review the code / bring it up to Oracle standards
  - Oracle finalized acquisition of Sun Microsystems in Jan 2010
  - Java SE 7 GA in Jul 2011

VENDORS RELATED ISSUES

Bugs in new features

"The perception is that these are new issues. Most of these are problems with JDK 1.4 and earlier„

Oracle VP, September 2013

VENDORS RELATED ISSUES

Bugs in new features (2)

- Almost all vulnerabilities we reported in 2012 / 2013 had its origin in classes introduced / modified in Java SE 7
  - `com.sun.org.glassfish.*` (Issues 1-7)
  - Beans decoder (Issues 11, 16, 17 and 28)
    - Bugs introduced in Java SE 7 implementation
  - Bytecode Verfifier issue (Issue 10)
- Multiple vulnerabilities in Method Handles API
  - Access to restricted classes (Issues 13 and 21)
  - Access to package scoped classes (Issue 26)
  - Bypass of caller based security checks (Issue 32)
- Java Control Panel security levels bypass
  - Click2Play bypass via serialized Applet instance (Issue 53)

VENDORS RELATED ISSUES

Incomplete patches

"I do not need you to analyze the code since we already do that, it's our job to do that, we are pretty good at it„

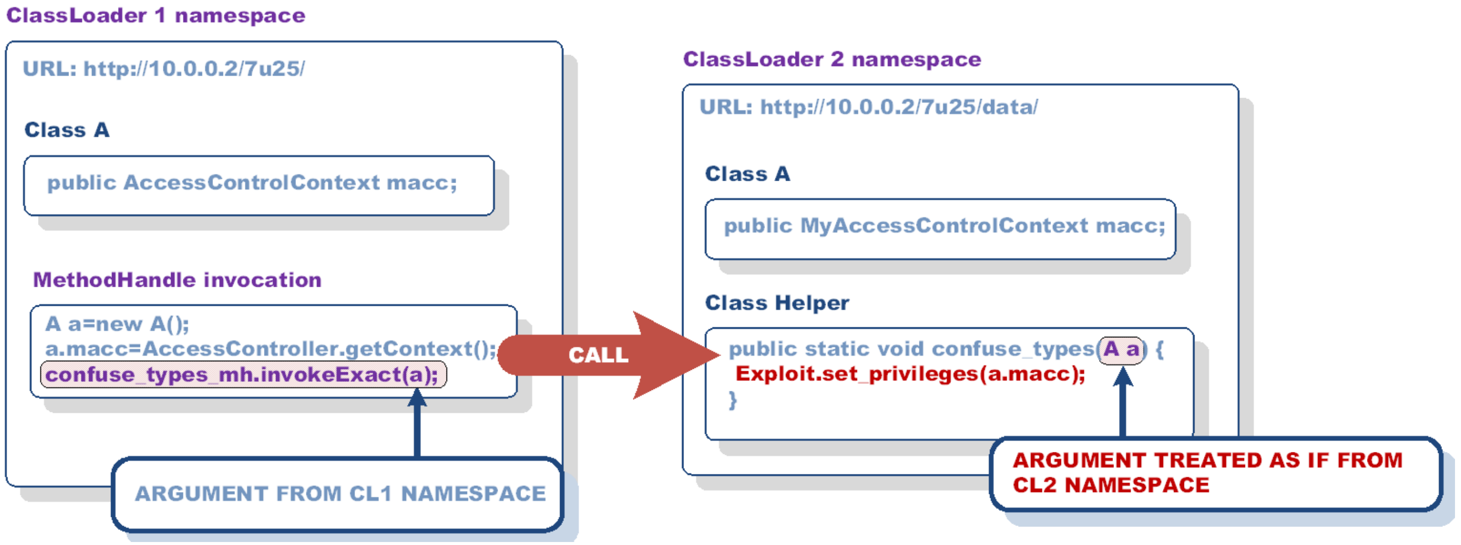
„please do not waste our time on reporting little green men in our code"

Oracle Chief Security Officer, Aug 2015

VENDORS RELATED ISSUES

Incomplete patches (Issue 69)

- Class spoofing attack vulnerability (Jul 2013)


ClassLoader 1 namespace
URL: http://10.0.0.2/7u25/
Class A
MethodHandle invocation
CALL
ClassLoader 2 namespace
URL: http://10.0.0.2/7u25/data/
Class A
Class Helper
ARGUMENT FROM CL1 NAMESPACE
ARGUMENT TREATED AS IF FROM CL2 NAMESPACE

```
public AccessControlContext macc;
```

```
A a=new A();
a.macc=AccessController.getContext();
confuse_types_mh.invokeExact(a);
```

```
public MyAccessControlContext macc;
```

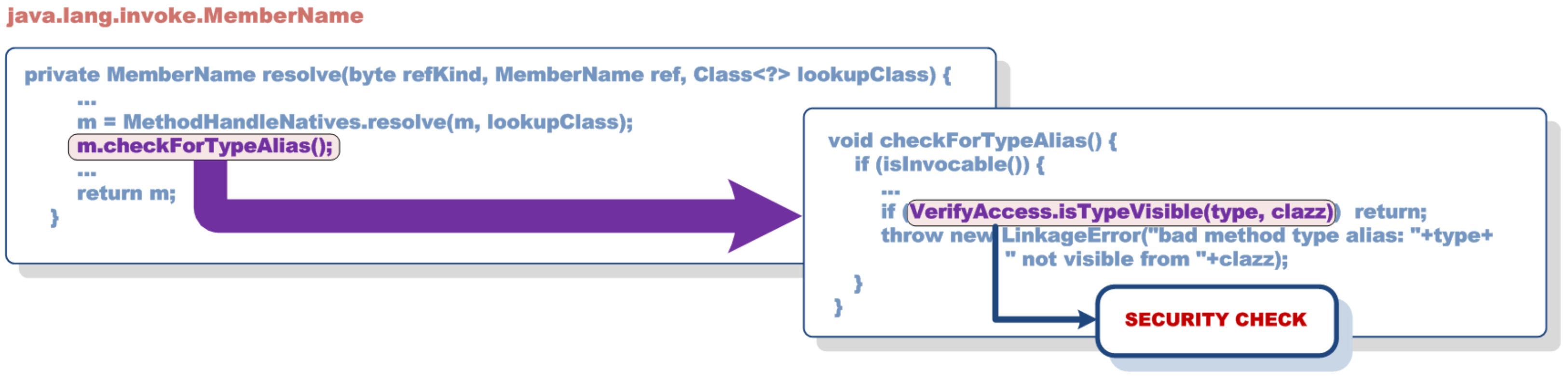
```
public static void confuse_types(A a) {
 Exploit.set_privileges(a.macc);
}
```

Original report: http://www.security-explorations.com/materials/SE-2012-01-ORACLE-13.pdf

VENDORS RELATED ISSUES

Incomplete patches (Issue 69 fix)

- Oracle "addressed" the problem in Java 7 Update 40
  - backport (from JDK 8) implementation of the affected component (method handles API)
  - a check for type aliasing (spoofing)



java.lang.invoke.MemberName

```
private MemberName resolve(byte refKind, MemberName ref, Class<?> lookupClass) {
      ...
      m = MethodHandleNatives.resolve(m, lookupClass);
      m.checkForTypeAlias();
      ...
      return m;
   }
```

```
void checkForTypeAlias() {
   if (isInvocable()) {
      ...
      if (VerifyAccess.isTypeVisible(type, clazz))  return;
      throw new LinkageError("bad method type alias: "+type+
                   " not visible from "+clazz);
   }
}
```

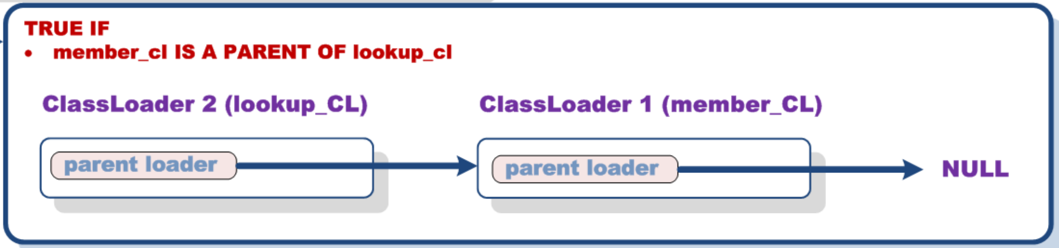
VENDORS RELATED ISSUES

Incomplete patches (Issue 69 fix cont.)

- Type visibility check

sun.invoke.util.VerifyAccess

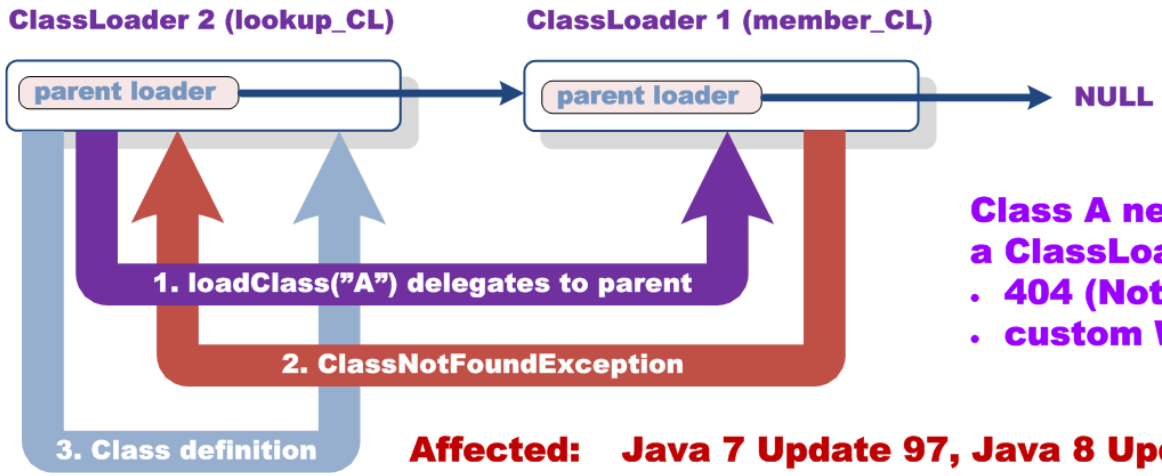
```
public static boolean isTypeVisible(Class memberType, Class lookupClass) {
    ...
    ClassLoader member_cl = memberType.getClassLoader();

    ClassLoader lookup_cl = lookupClass.getClassLoader();
    ...

    if (member_cl == lookup_cl ||(loadersAreRelated(member_cl, lookup_cl, true)))
        return true;
    ...
```


TRUE IF
• member_cl IS A PARENT OF lookup_cl
ClassLoader 2 (lookup_CL)
parent loader
ClassLoader 1 (member_CL)
parent loader
NULL

VENDORS RELATED ISSUES

Incomplete patches (Issue 69 fix bypass)

- Custom Class Loader setup results in a trivial bypass of Oracle patch
  - A change of 4 characters in original POC code from 2013


ClassLoader 2 (lookup_CL)
ClassLoader 1 (member_CL)
parent loader
parent loader
NULL
1. loadClass("A") delegates to parent
2. ClassNotFoundException
3. Class definition

Class A needs to be defined in a ClassLoader 2 namespace
- **404 (Not Found) trick**
- **custom WWW server**

Affected: Java 7 Update 97, Java 8 Update 74, Java 9 EA 108

VENDORS RELATED ISSUES
Incomplete patches (others)

- IBM struggled to properly fix some Java vulnerabilities as well
  - Issues 35, 36 and 37 (improperly patched)
    - insecure use of `invoke` method of `java.lang.reflect.Method` class
  - Issue 49 (improperly patched for two times)
    - insecure use of `defineClass` method of `java.lang.ClassLoader` class
- Fix addressing scenarios illustrated by Proof of Concept codes
  - Lack of understanding of the actual root cause of the reported issues

VENDORS RELATED ISSUES

Ignorance of vendor's recommendations

"Ultimately, people have to update"

Oracle VP, May 2013

VENDORS RELATED ISSUES

Ignorance of vendor's recommendations (2)

- Oracle Java Cloud Service
  - `java.runtime.version=1.6.0_37-b06` (EMEA1 data center)
  - `java.runtime.version=1.7.0_15-b33` (US1 data center)
  - approx. 150 security fixes incorporated into Java SE software since the end of 2012 / beginning of 2013 were missing from the environment
- Oracle Database Java VM
  - No regular Java VM updates prior to Oct 2014
    - Would Oracle start these updates if Security Explorations hasn't shown that their flagship Database could be hacked with Java ?

VENDORS RELATED ISSUES

Ignorance of vendor's recommendations (3)

- Google App Engine
  - 1+ years old JRE (< JDK 7 Update 40)
  - approx. 100 security fixes incorporated into Java SE software since the end of 2012 / beginning of 2013 were missing from the environment
    - could be successfully hacked with the use of a public vulnerability / exploit code from Oct 2013 till Mar 2015
    - this could be achieved regardless of Google's mitigations preventing known Java vulnerabilities from being exploitable

VENDORS RELATED ISSUES

Ignorance of own Secure Coding Guidelines

"The implementation of Oracle Software Security Assurance policies and practices by Java development is also intended to defend against the introduction of new vulnerabilities into the Java code base"

Oracle VP, May 2013

"And our goal is that there will be no, zero, absolutely none, no security vulnerabilities in Java"

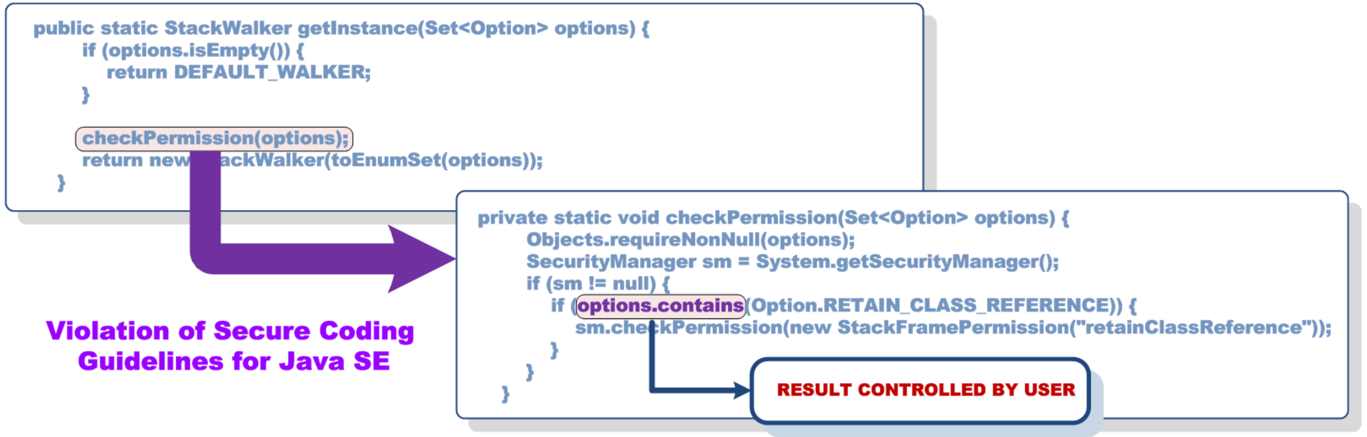
Oracle VP, September 2013

VENDORS RELATED ISSUES

Ignorance of own Secure Coding Guidelines (2)

- JDK 9 Early Access Build 108 (Mar 04, 2016)

java.lang.StackWalker



```
public static StackWalker getInstance(Set<Option> options) {
        if (options.isEmpty()) {
            return DEFAULT_WALKER;
        }

        checkPermission(options);
        return new [illegible]ackWalker(toEnumSet(options));
    }
```

```
private static void checkPermission(Set<Option> options) {
        Objects.requireNonNull(options);
        SecurityManager sm = System.getSecurityManager();
        if (sm != null) {
            if (options.contains(Option.RETAIN_CLASS_REFERENCE)) {
                sm.checkPermission(new StackFramePermission("retainClassReference"));
            }
        }
    }
```

Violation of Secure Coding Guidelines for Java SE

Guideline 6-3 / MUTABLE-3: Create safe copies of mutable and subclassable input values

„a time-of-check, time-of-use inconsistency (TOCTOU) [7] can be exploited where a mutable input contains one value during a SecurityManager check but a different value when the input is used later

VENDORS RELATED ISSUES

Long patch times / cycles

- Most vendors address security issues promptly only when they are actively exploited
  - Issues addressed by Oracle Security Alert from Aug 28, 2012 were known to the vendor for 4 months
- Critical security vulnerabilities should be patched as soon as possible
  - Minimizing the risk of
    - a discovery and attacks by other parties
    - a leak / theft from the vendor
  - Passing the message to the world that security is a priority
- It's the matter of the engineering resources that are put towards security

VENDORS RELATED ISSUES
Long patch times / cycles (notable case)

- A fix for a critical vulnerability affecting all Java versions
  - Original report in Sep 2012
  - Fix in Jan 2013 (4 months)
  - Patch could be implemented within 28 minutes
    - Oracle patch was a mirrored version of the fix we proposed

VENDORS RELATED ISSUES

Inaccurate / untrue / deceptive statements

„the security problems affecting Java in Internet browsers have generally not impacted Java running on servers”

Oracle VP, May 2013

VENDORS RELATED ISSUES

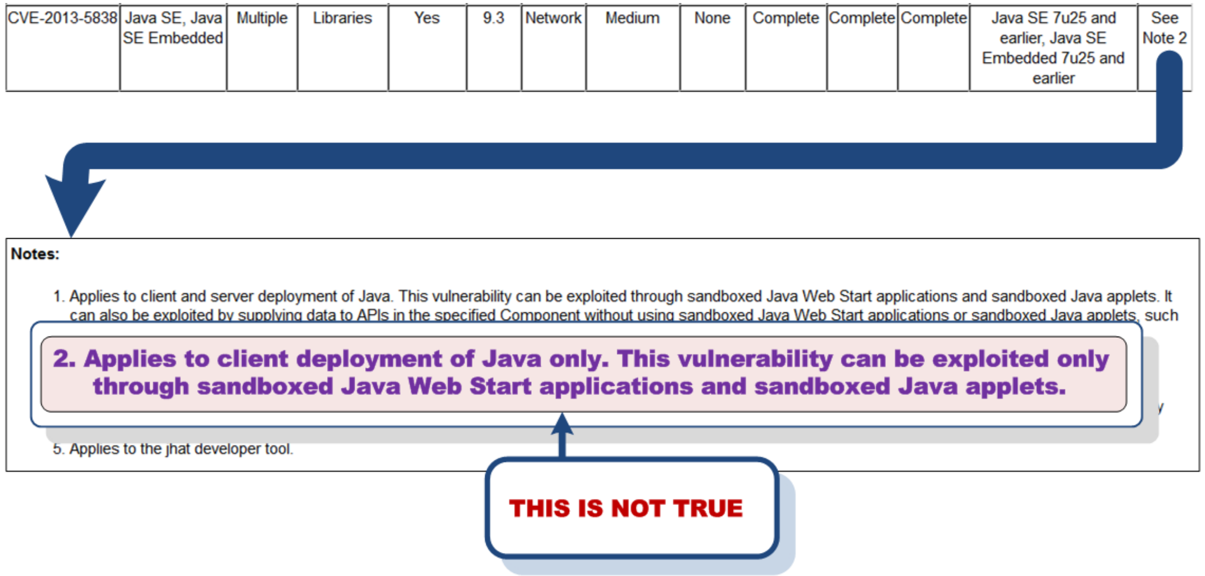
Inaccurate / untrue / deceptive statements (2)

- Oracle provides misleading and inaccurate information about the impact of Java Security Vulnerabilities
  - Downplaying vulnerability impact by indicating that it affects clients / Java Plugin only (not servers)
  - Server deployments also vulnerable
    - Server JRE
    - RMI servers
    - GlassFish server
    - Weblogic server
    - Cloud environments
    - Oracle Database
    - …

VENDORS RELATED ISSUES

Inaccurate / untrue / deceptive statements (3)

- CVE-2013-5838 (Issue 69, SE-2012-01)
  - Successfully exploited in server environment (GAE)



CVE-2013-5838	Java SE, Java SE Embedded	Multiple	Libraries	Yes	9.3	Network	Medium	None	Complete	Complete	Complete	Java SE 7u25 and earlier, Java SE Embedded 7u25 and earlier	See Note 2

Notes:

1. Applies to client and server deployment of Java. This vulnerability can be exploited through sandboxed Java Web Start applications and sandboxed Java applets. It can also be exploited by supplying data to APIs in the specified Component without using sandboxed Java Web Start applications or sandboxed Java applets, such

2. Applies to client deployment of Java only. This vulnerability can be exploited only through sandboxed Java Web Start applications and sandboxed Java applets.

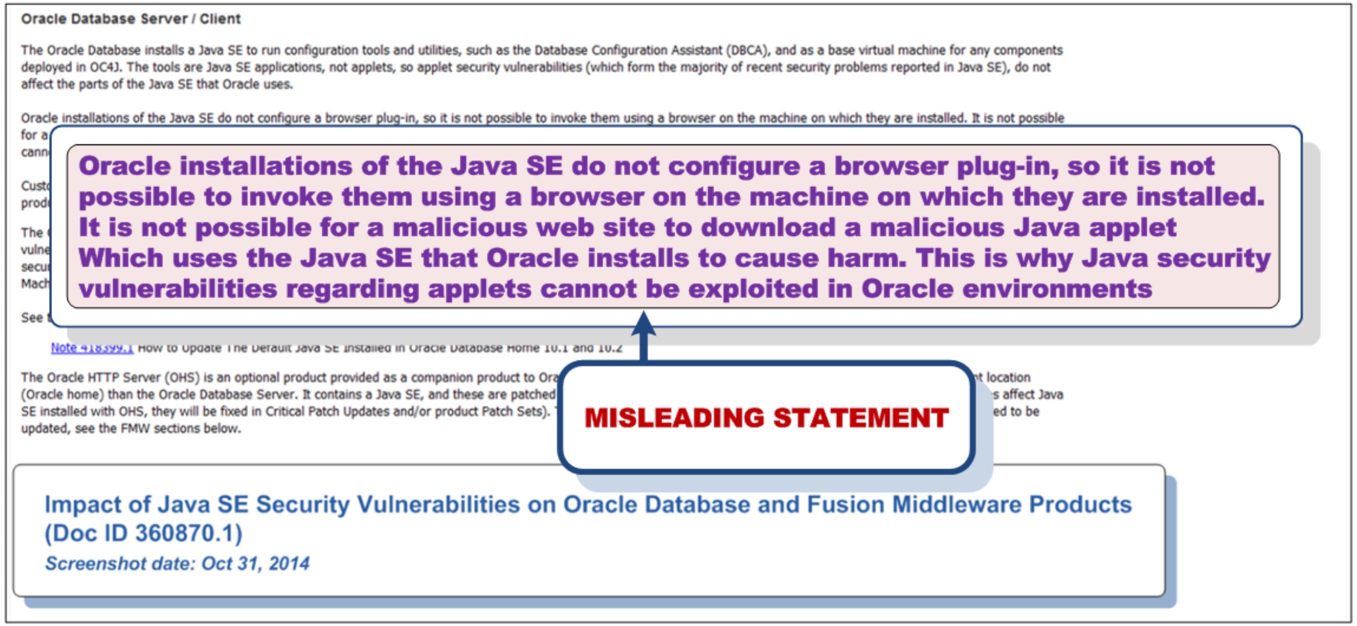
5. Applies to the jhat developer tool.

THIS IS NOT TRUE

VENDORS RELATED ISSUES

Inaccurate / untrue / deceptive statements (4)

- Oracle Support Document 360870.1



Oracle Database Server / Client

The Oracle Database installs a Java SE to run configuration tools and utilities, such as the Database Configuration Assistant (DBCA), and as a base virtual machine for any components deployed in OC4J. The tools are Java SE applications, not applets, so applet security vulnerabilities (which form the majority of recent security problems reported in Java SE), do not affect the parts of the Java SE that Oracle uses.

Oracle installations of the Java SE do not configure a browser plug-in, so it is not possible to invoke them using a browser on the machine on which they are installed. It is not possible for a
cann

Custo
prod

The
vulne
secu
Mach

See t

Note 418399.1 How to Update The Default Java SE Installed in Oracle Database Home 10.1 and 10.2

The Oracle HTTP Server (OHS) is an optional product provided as a companion product to Ora … t location (Oracle home) than the Oracle Database Server. It contains a Java SE, and these are patched … s affect Java SE installed with OHS, they will be fixed in Critical Patch Updates and/or product Patch Sets). … ed to be updated, see the FMW sections below.

Oracle installations of the Java SE do not configure a browser plug-in, so it is not possible to invoke them using a browser on the machine on which they are installed. It is not possible for a malicious web site to download a malicious Java applet Which uses the Java SE that Oracle installs to cause harm. This is why Java security vulnerabilities regarding applets cannot be exploited in Oracle environments

MISLEADING STATEMENT

Impact of Java SE Security Vulnerabilities on Oracle Database and Fusion Middleware Products (Doc ID 360870.1)

Screenshot date: Oct 31, 2014

VENDORS RELATED ISSUES

Playing with bug numbers

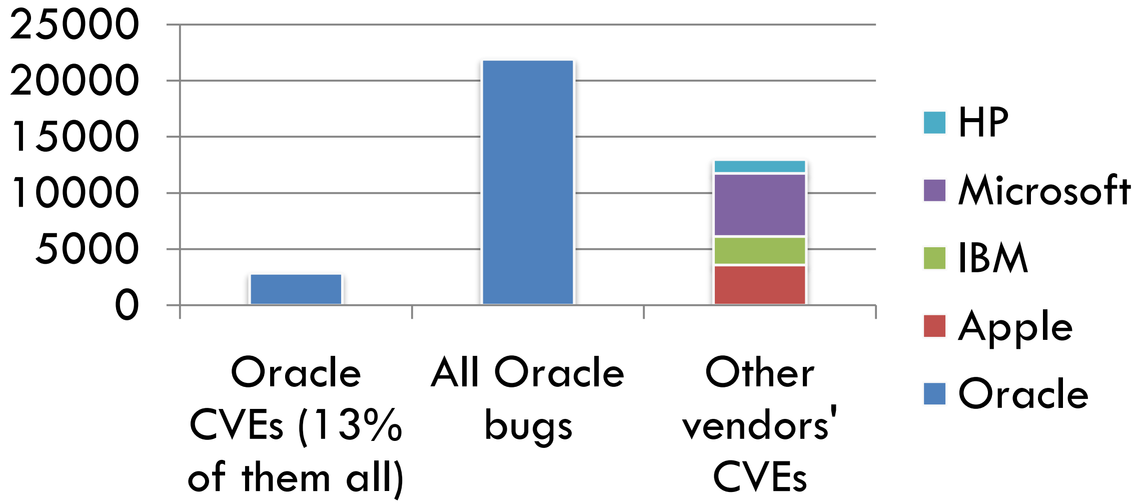
„Ah, well, we find 87% of security vulnerabilities ourselves, security researchers find about 3% and the rest are found by customers„

Oracle Chief Security Officer, Aug 2015

VENDORS RELATED ISSUES

Playing with bug numbers (2)

- Implication that Oracle finds twice as many security bugs as Microsoft, IBM, HP and Apple issues combined together



VENDORS RELATED ISSUES

Playing with bug numbers (3)

- IBM and Oracle tend to cumulate multiple different security issues under one CVE identifier
  - The number of security fixes announced by the companies do not necessarily reflect the number of real security issues addressed

VENDOR	CVE ID	# DIFFERENT ISSUES / CODE LOCATIONS
ORACLE	CVE-2012-5076	Issues 1-7 (**7**)
ORACLE	CVE-2012-4681	Issues 11, 16 and 17 (**3**)
IBM	CVE-2012-4820	Issues 33-37 (**5**)
IBM	CVE-2012-4822	Issues 40-48 (**9**)

VENDORS RELATED ISSUES

Lack of understanding of Java security model

- Our experience is that all vendors had problems with it
  - Apple
    - no understanding that modern Java security exploits may combine several innocent looking bugs
  - IBM
    - really simple bugs, bugs in custom Java API / VM implementation
  - Oracle
    - mixing of Oracle Database and Java VM security models
      - Oracle Database security weakened!

VENDORS RELATED ISSUES

Lack of understanding of Java security model (GAE case)

- GAE implements additional restricted classes namespace on top of JRE, but it does not implement security checks in all locations where such classes could be referenced
  - security checks related to the class linking and methods resolution
- As a result, user defined classes could be linked with restricted GAE classes
  - they could subclass from them and call their methods via `invokevirtual` / `invokespecial` / `invokestatic` bytecode instructions
  - Issue 40 (not accepted by Google)

VENDORS RELATED ISSUES
Disrespect to security researchers

- Security researchers are one of driving forces for the infosec industry
  - Vulnerabilities discovery, novel attack /exploitation techniques
- Many of them work for free and do not expect much back
  - Vulnerability confirmation, status updates, prompt fix, credits
- Not all companies appreciate this contribution
  - Oracle CSO blog post from Aug 2015
    - Oracle officially distancing from it, but CSO not changed
      - Should be replaced by Sun CSO (Whitfield Diffie) years ago
  - Problems during vulnerability handling
    - Not responding to e-mails, not providing status updates, neglecting to confirm reported issues, silent fixes, no credits, …

VENDORS RELATED ISSUES

Disrespect to security researchers (2)

- The vulnerability handling process is the best illustration of vendors attitude towards security researchers

Vendor	Target	#Issues	Vuln. Handling	Reward
ADB	STB SW	20	Problems (no response / complete drop of communication)	No
ORACLE	Java SE	44	Acceptable	No
ORACLE	Cloud	30	Problems (forced disclosure)	No
ORACLE	DB JVM	22	Acceptable	No
IBM	Java SE	26	Acceptable	No
APPLE	Java SE	2	Problems (silent fix)	No
GOOGLE	Cloud	41	Acceptable / problems (forced disclosure)	$100k

ECOSYSTEM ISSUES
No info sharing

- There are many parties that rely on Oracle when it comes to vulnerability / fix information
  - Other vendors cannot release Java patches until Oracle shares their details
  - Java licensees likely have their hands tied with EULA's / legal statements
- Oracle policy to not discuss security vulnerabilities' details seems to cause more harm than good
  - others look for truly basic vulnerability impact information
    - Antivirus vendors, nuclear plants administrators, cloud service providers, …
  - malicious parties can reverse engineer patches anyway
  - security awareness not increased
    - what pitfalls to avoid / what to look for during code development / security review efforts

SIGNATURE PROJECT
SE-2014-02

- Google App Engine Java security sandbox bypasses
  - Security research project evaluating security of Java security sandbox used in Google Cloud environment
    - 40+ security issues / 30 Proof of Concept codes reported to Google
- It perfectly illustrates many of the issues outlined in this presentation
  - Complexity of Java security
  - No deep understanding of a Java security model
  - Ignorance of vendor's guidelines / recommendations
  - Little Java exploit mitigations
  - …

Project report: http://www.security-explorations.com/materials/se-2014-02-report.pdf

SIGNATURE PROJECT

Google conclusions

- The company concluded that its best to treat Java VM as compromised and focus on building additional sandboxing layer around it

*„it's really *really* hard to do what we want to do in Java-world securely if we want to support reflection and other crazy things outside of the traditional Java Security Model"*

SUMMARY
Final Words

- The language that was supposed to be a secure alternative to C / C++ is alone full of security hazards
- Oracle gross incompetence in SW security assurance has lead to the lost of trust of businesses and consumers
- Company's untrue, misleading and deceptive statements regarding Java security only worsened the situation
- The decision to denounce the primary security feature of Java was one of the worst possible

SUMMARY
Final Words (cont.)

- The whole SW industry needs to be taken with caution
  - little liabilities
  - customers have no choice than to trust vendors' claims
    - closed, proprietary software
    - clouds environments
- Independent security evaluations showing the real state of software security are an important part of the equation
- Government authorities putting vendors to order over poor / deceptive security practices can pave the way for SW liabilities
- Ecosystem works better if key players are not afraid to say enough!
  - Web browser vendors

FINAL

Q & A

THANK YOU

contact@security-explorations.com