Anatomy of Java Vulnerabilities Java2Days 2017

Steve Poole

@spoole167

About me

Steve Poole

IBM Lead Engineer / Developer advocate

Making Java Real Since Version 0.9

Open Source Advocate

DevOps Practitioner (whatever that means!)

Driving Change

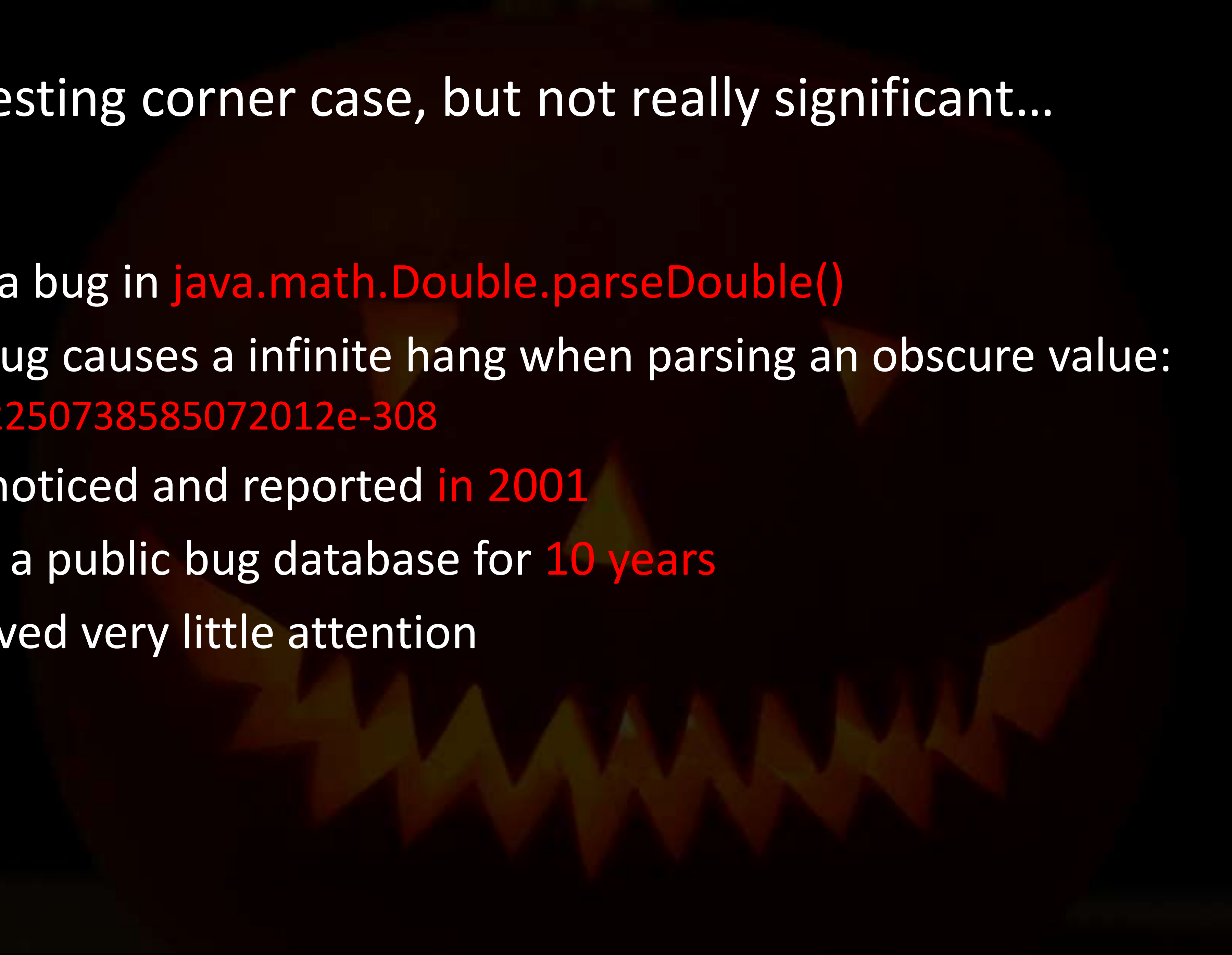
@spoole167

Consider this two character change

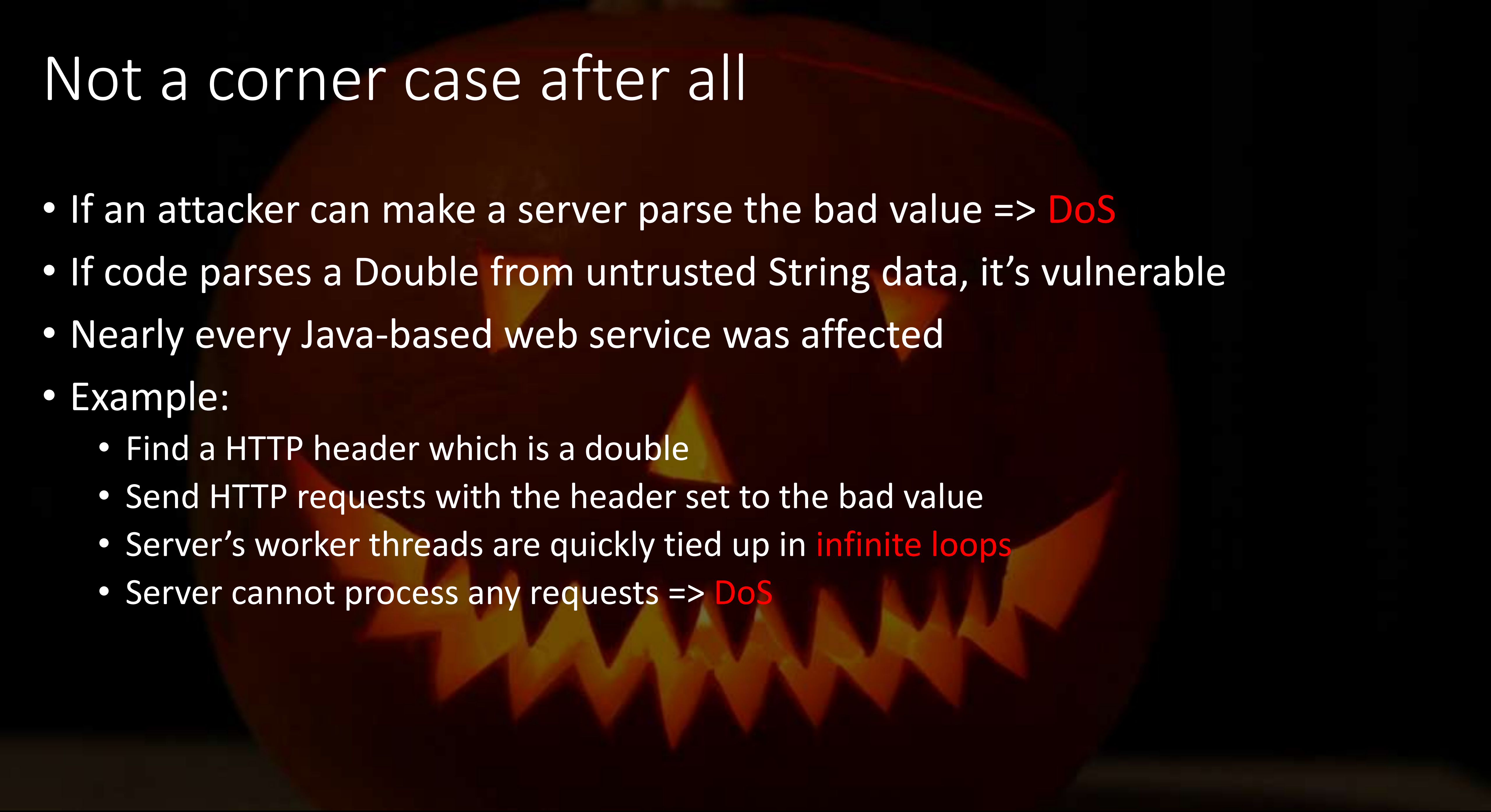
```
@@ -1529,7 +1527,7 @@ public class FloatingDecimal{
                if ( (cmpResult = bigB.cmp( bigD ) ) > 0 ){
                    overvalue = true; // our candidate is too big.
                    diff = bigB.sub( bigD );
-                   if ( (bigIntNBits == 1) && (bigIntExp > -expBias) ){
+                   if ( (bigIntNBits == 1) && (bigIntExp > -expBias+1) ){
                        // candidate is a normalized exact power of 2 and
                        // is too big. We will be subtracting.
                        // For our purposes, ulp is the ulp of the
```

@spoole167

Interesting corner case, but not really significant…



- Fixes a bug in java.math.Double.parseDouble()
- The bug causes a infinite hang when parsing an obscure value:
  - 2.2250738585072012e-308
- First noticed and reported in 2001
- Sat in a public bug database for 10 years
- Received very little attention



Not a corner case after all

- If an attacker can make a server parse the bad value => DoS
- If code parses a Double from untrusted String data, it’s vulnerable
- Nearly every Java-based web service was affected
- Example:
  - Find a HTTP header which is a double
  - Send HTTP requests with the header set to the bad value
  - Server’s worker threads are quickly tied up in infinite loops
  - Server cannot process any requests => DoS

Rediscovered in 2011:

http://www.exploringbinary.com/java-hangs-when-converting-2-2250738585072012e-308/

Trivial to exploit + Curious people everywhere = Huge impact

Large cost to everyone to produce and apply patches

So many servers were brought to their knees ‘just for kicks’

This talk is a technical horror story

https://www.flickr.com/photos/koolmann/

What is a Vulnerability?

“A vulnerability is a bug which can be exploited by an attacker”

Exploits are many and various

- Exploits can attack your availability
  - Bringing your system down by making it crash
  - Making your system unresponsive though excessive memory / CPU / network usage
- Exploits can reduce Integrity
  - Modification of application data
  - Arbitrary code execution
- Exploits can breech confidentiality
  - Privilege elevation
  - Exposure of sensitive information

Why should you care?

Cybercrime realities

https://www.flickr.com/photos/koolmann/

Do you think cybercriminals are lone hackers?

https://www.flickr.com/photos/stignygaard/

Organized Cybercrime is the most profitable type of crime

- In 2016 Cybercrime was estimated to be worth 445 Billion Dollars a Year

- In 2013 the United Nations Office on Drugs and Crime (UNODC) estimated globally the illicit drug trade was worth 435 Billion Dollars

- Guess which one has the least risk to the criminal ?
- Guess which is growing the fastest ?
- Guess which one is the hardest to prosecute ?

- Guess which one is predicted to reach 2100 Billion Dollars by 2019?

Another vulnerability

I have this directory…

“/Users/spoole/foo bar” ← with a space in the name

I want to refer to it in a URL….

"file:///Users/spoole/foo%20bar"

@spoole167

```
URL u=new URL("file:///Users/spoole/foo%20bar");

File f=new File(u.getPath());

System.out.println("path="+f.getAbsolutePath());
System.out.println("exists="+f.exists());
```

```
URL u=new URL("file:///Users/spoole/foo%20bar");

File f=new File(u.getPath());

System.out.println("path="+f.getAbsolutePath());
System.out.println("exists="+f.exists());
```

```
path=/Users/spoole/foo%20bar
exists=false
```

Oops – forgot to decode it

```
URL u=new URL("file:///Users/spoole/foo%20bar");

URI uri = u.toURI();

File f=new File(uri.getPath());

System.out.println("path="+f.getAbsolutePath());

System.out.println("exists="+f.exists());
```

```java
URL u=new URL("file:///Users/spoole/foo%20bar");

URI uri = u.toURI();

File f=new File(uri.getPath());

System.out.println("path="+f.getAbsolutePath());

System.out.println("exists="+f.exists());
```

```
path=/Users/spoole/foo bar
exists=true
```

What would happen if someone had created a directory called

”/Users/spoole/foo%20bar” ?

What would happen if someone had created a directory called

"/Users/spoole/foo%20bar" ?

```
path=/Users/spoole/foo%20bar
exists=true
```

@spoole167

Not a big deal?

@spoole167

On Windows ***anyone*** can create a top level directory

```
md “C:\foo%20bar”
```

@spoole167

Still not a big deal?

@spoole167

What happens if your extensions path has an entry like

“C:/Program%20Files/optional-product/extensions”

And you didn’t have the product installed.

You might never notice

@spoole167

Suppose the JVM didn’t decode the entry when searching for dll’s etc

And suppose someone created a directory that matched “C:/Program%20Files/optional-product/extensions”

Someone might be able to get your application to load sinister dlls

Because of 1 line of missing code:

```
URI uri = u.toURI();
```

@spoole167

Not quite a true story.
But a similar vulnerability
has been fixed in the JVM

https://www.flickr.com/photos/koolmann/

Vulnerabilities are almost always simple
there are no smoking guns
Exploits are chaining together vulnerabilities

Who’s being targeted?

- Middle level executives – afraid of their bosses?
- New joiners – easy to make a mistake?
- Busy and harassed key individuals – too busy to take time to consider?
- Disgruntled employees – want to hurt the company? Make some $?

- And Developers – the golden goose.

The bad guys prey on the weak, vulnerable and ignorant

Developers

- Why ?
  - We know the inside story
  - We write the code
  - We have elevated privileges
  - We are over trusting
  - We use other peoples code and tools without inspection
  - we are ignorant of security matters

The bad guys prey on the weak, vulnerable and ignorant

The bad guys prey on the weak, vulnerable and ignorant:

That’s us

Don’t agree?

Ever googled for:

“Getting Java to accept all certs over HTTPS”

“How to Trust Any SSL Certificate”

“very trusting trust manager”

“Disable Certificate Validation in Java”

Ever written something like this?

```
TrustManager[] trustAllCerts = new TrustManager[]{

        new X509TrustManager() {
            public X509Certificate[] getAcceptedIssuers() {
                return null;
            }
            public void checkClientTrusted(
                X509Certificate[] certs, String authType) {
            }
            public void checkServerTrusted(
                X509Certificate[] certs, String authType) {
            }
            public boolean isClientTrusted( X509Certificate[] cert) {
                return true;
            }
            public boolean isServerTrusted( X509Certificate[] cert) {
                return true;
            }
        }}
```

We’ve all done something like that

We’ve all done something like that

We do it all the time

We’ve all done something like that

We do it all the time

The whole world does it

How bad can it be?

We’ve all done something like that

We do it all the time

The whole world does it

Github search “implements TrustManager” ….

AllTrustingSecurityManagerPlugin.java

We’ve found 72,609 code results

```
// Install the all-trusting trust
manager
```

```
OverTrustingTrustProvider
```

```
AlwaysValidTrustManager
```

```
A very trusting trust manager that
accepts anything
```

AllTrustingCertHttpRequester.java

AcceptingTrustManagerFactory.java

```
TrustAllServersWrappingTrustManager
```

```
A very friendly, accepting trust
manager factory. Allows anything
through.
```

```
all kind of certificates are
accepted and trusted.
```

We’ve all done something like that
Sometimes it even a ‘feature’

"A vulnerability is a bug which can be exploited by an attacker"

“A vulnerability is a bug which can be exploited by an attacker”

“A vulnerability is also a feature which can be exploited by an attacker”

Vulnerabilities

- Bugs and design flaws in your software and the software you use.
- Everyone has them.

- Researchers are looking for them all the time.
- **So are the bad guys**

https://www.flickr.com/photos/electronicfrontierfoundation/

The process of managing vulnerabilities

“Common Vulnerabilities & Exposures”

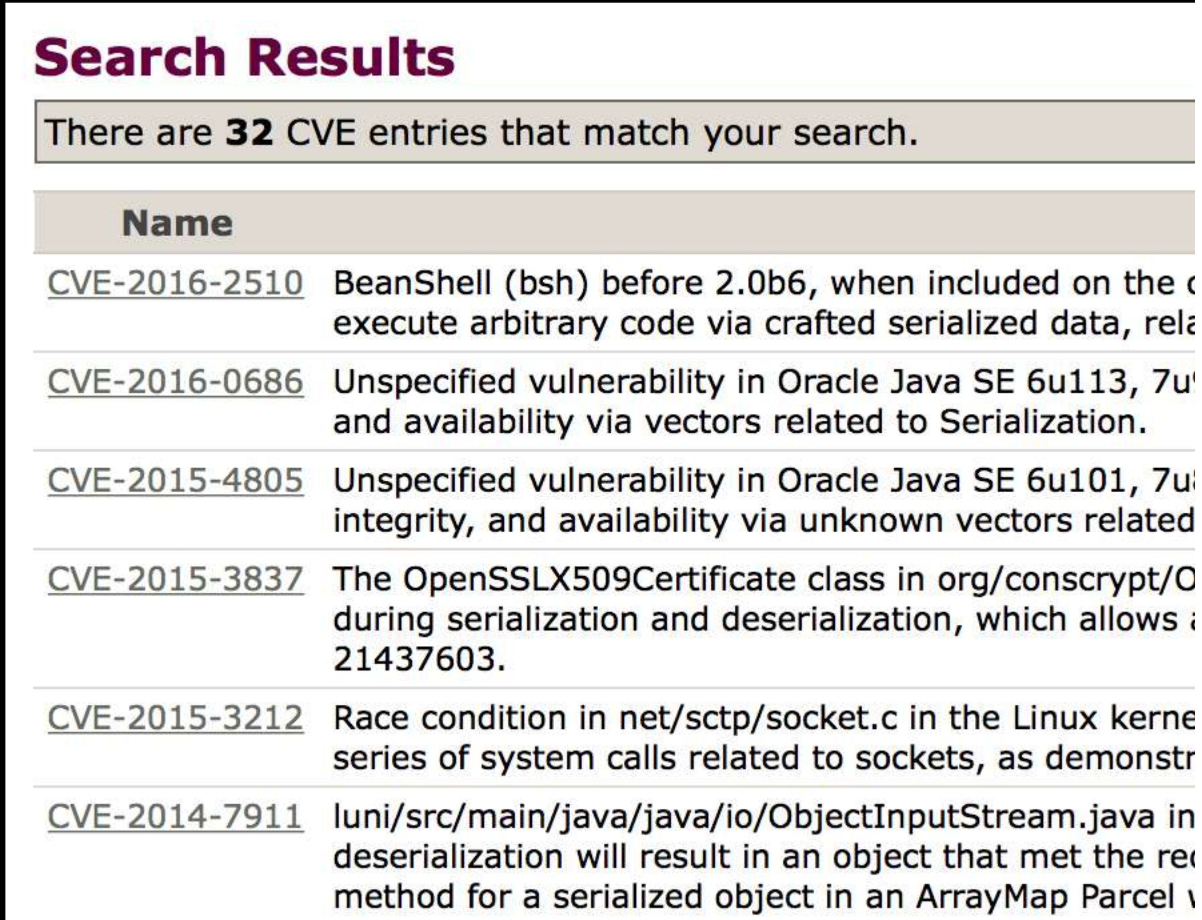
- https://cve.mitre.org

- The Standard place find details about ‘CVEs’
- International cyber security community effort
- Common naming convention and unique references.
- Allows you to know when a problem is resolved in something you are using

@spoole167

‘CVEs’ https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=*

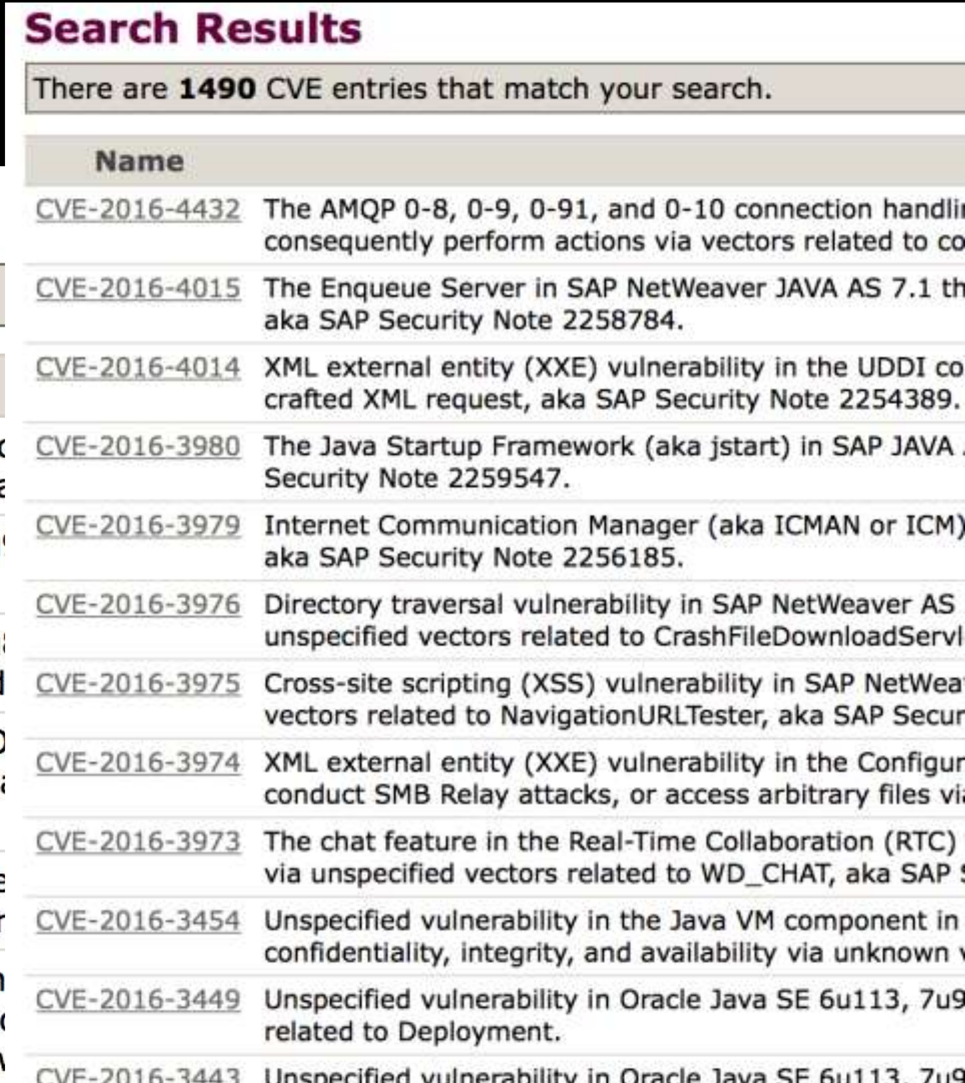
keyword=java “**1662**”



Search Results

There are **32** CVE entries that match your search.

Name	
CVE-2016-2510	BeanShell (bsh) before 2.0b6, when included on the c execute arbitrary code via crafted serialized data, rela
CVE-2016-0686	Unspecified vulnerability in Oracle Java SE 6u113, 7u and availability via vectors related to Serialization.
CVE-2015-4805	Unspecified vulnerability in Oracle Java SE 6u101, 7u integrity, and availability via unknown vectors related
CVE-2015-3837	The OpenSSLX509Certificate class in org/conscrypt/O during serialization and deserialization, which allows a 21437603.
CVE-2015-3212	Race condition in net/sctp/socket.c in the Linux kerne series of system calls related to sockets, as demonstr
CVE-2014-7911	luni/src/main/java/java/io/ObjectInputStream.java in deserialization will result in an object that met the rec method for a serialized object in an ArrayMap Parcel w



Search Results

There are **1490** CVE entries that match your search.

Name	
CVE-2016-4432	The AMQP 0-8, 0-9, 0-91, and 0-10 connection handli consequently perform actions via vectors related to co
CVE-2016-4015	The Enqueue Server in SAP NetWeaver JAVA AS 7.1 th aka SAP Security Note 2258784.
CVE-2016-4014	XML external entity (XXE) vulnerability in the UDDI co crafted XML request, aka SAP Security Note 2254389.
CVE-2016-3980	The Java Startup Framework (aka jstart) in SAP JAVA Security Note 2259547.
CVE-2016-3979	Internet Communication Manager (aka ICMAN or ICM) aka SAP Security Note 2256185.
CVE-2016-3976	Directory traversal vulnerability in SAP NetWeaver AS unspecified vectors related to CrashFileDownloadServl
CVE-2016-3975	Cross-site scripting (XSS) vulnerability in SAP NetWea vectors related to NavigationURLTester, aka SAP Secur
CVE-2016-3974	XML external entity (XXE) vulnerability in the Configur conduct SMB Relay attacks, or access arbitrary files vi
CVE-2016-3973	The chat feature in the Real-Time Collaboration (RTC) via unspecified vectors related to WD_CHAT, aka SAP S
CVE-2016-3454	Unspecified vulnerability in the Java VM component in confidentiality, integrity, and availability via unknown v
CVE-2016-3449	Unspecified vulnerability in Oracle Java SE 6u113, 7u9 related to Deployment.
CVE-2016-3443	Unspecified vulnerability in Oracle Java SE 6u113, 7u9

keywords=java serialization (first three as of May 11th)

CVE-2016-2510	BeanShell (bsh) before 2.0b6, when included on the classpath by an application that uses Java serialization or XStream, allows remote attackers to execute arbitrary code via crafted serialized data, related to XThis.Handler.
CVE-2016-0686	Unspecified vulnerability in Oracle Java SE 6u113, 7u99, and 8u77 and Java SE Embedded 8u77 allows remote attackers to affect confidentiality, integrity, and availability via vectors related to Serialization.
CVE-2015-4805	Unspecified vulnerability in Oracle Java SE 6u101, 7u85, and 8u60, and Java SE Embedded 8u51, allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors related to Serialization.

@spoole167

CVE-2016-0686

“Unspecified vulnerability in Oracle Java SE 6u113, 7u99, and 8u77 and Java SE Embedded 8u77 allows remote attackers to affect confidentiality, integrity, and availability via vectors related to Serialization.”

That’s all you will find about this fix

Talking about the details of a fix or flaw in public is just like tweeting your credit card # and pin

So we don’t

We give you information about the impact

Common Vulnerability Scoring System

- Base CVSS combines several aspects into a single score
  - Attack vector and complexity – local, remote etc.
  - Impact – integrity, confidentiality etc.
  - Privileges required?
- Represented as a base score and a CVSS *vector*
  - CVSS 9.6 (CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:H)
- Additional temporal CVSS considers aspects that may change over time
  - Availability of an exploit
  - Availability of a fix
- IBM X-Force
  - IBM's security research organization
  - Provides base and temporal CVSS scores for **all** published CVEs

Example CVSS vector

- CVSS Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:H
    - **A**ttack **V**ector: **N** (network)
    - **A**ttack **C**omplexity: **L** (low)
    - **P**rivileges **R**equired: **N** (none)
    - **U**ser **I**nteraction: **R** (required)
    - **S**cope: **C** (changed)
    - **C**onfidentiality impact: **H** (high)
    - **I**ntegrity impact: **H** (high)
    - **A**vailability impact: **H** (high)

Note – assessments are based on the assumption that everyone behaves "sensibly"
If you give everyone root access to a machine your mileage will differ

- Using the CVSS v3.0 calculator
- Low = 0.0-3.9. Medium= 4.0-6.9 High= 7.0-8.9 Critical = 9.0-10.0

@spoole167

How are vulnerabilities communicated?

- Oracle
  - CPU Advisory
  - Risk Matrix
- IBM
  - Java and Product Security Bulletins
    - All IBM Security Bulletins are communicated on the PSIRT blog
    - Can subscribe to Bulletins for specific products using My Notifications
  - Java SDK Developer Centre

What if I discover a vulnerability?

- Report it **responsibly**

- Oracle or OpenJDK component (e.g. HotSpot VM, AWT, Net, Plugin)
    - Report it to Oracle

- IBM component (e.g. J9VM, IBM ORB, IBM JCE/JSSE)
    - Report it to IBM

- Dont
    - shout about it!
    - send a mail to an OpenJDK mailing list
    - post the details on a forum/blog
    - sell it to the bad guys
    - worry about apparent lack of severity – always report it

How are Java vulnerabilities addressed?

- Ideally as quickly as possible, but it depends on
  - CVSS
  - Impending disclosure
  - Resources

- In general:
  - Issues categorised internally
  - Fixes targeted for a future scheduled regular security release
  - Fixes may be deferred or brought forward if things change
  - Extremely urgent issues may trigger an out of cycle release

@spoole167

Vulnerability applicability

- High CVSS is irrelevant if the issue does not apply to you
  - Java plugin flaws don’t apply to appserver deployments
  - JAXP issues aren’t relevant if you don’t parse untrusted XML data
- Use the Oracle CPU Risk Matrix
  - Base CVSS scores and vectors
  - Consider the attack vector for the vulnerability
  - Oracle Applicability notes
- IBM customers: if in doubt, contact IBM Support
  - IBM Products have detailed applicability information

If you don’t know what your code does: you don’t know if you’re vulnerable

If you don’t know what your dependencies do: you don’t know if you’re vulnerable

BTW: You don’t know all your dependencies

Attack Vectors

@spoole167

Attack Vectors – Untrusted Code

- Code is able to bypass or escape the SecurityManager
- Affects any application which runs untrusted code under a security manager
  - Some server applications do this!
- Flaw can be in **any** component on the classpath
- **Partial bypass**
  - Exposure of sensitive information – e.g. system property values, heap contents
  - Arbitrary network connections
- **Full bypass**
  - Arbitrary code execution and access to operating system
  - Many server deployments run as root/admin

SecurityManager offers no protection against CPU DoS attacks or Infinite loops

Attack Vectors – Plugin / Web Start

- True “client-side” issues
- Affect JREs installed as the system default
- Exploits can be triggered remotely, usually via a browser
  - Malicious applet or JNLP file
- May involve platform or browser specific components
  - Installer/updater
  - Windows Registry entries
  - Browser callbacks
  - OS Shell behaviour (command line parsing)
  - Inappropriate privilege elevation
- Applets and browser plugins are deprecated in JDK 9

Attack Vectors – Untrusted Data

- A Java SE API mishandles a specific type of data
  - XML, JPG, URLs, Fonts, JARs, Strings
  - Issues tend to be more severe when the API is implemented natively
- Maliciously-crafted data can exploit the bug
  - DoS – CPU/memory usage, crash
  - Exposure of sensitive data – arbitrary memory access
  - Arbitrary code execution – code injection, disabling of SecurityManager
- Affects any application which handles the specific data type from untrusted sources
- Examples
  - A server application which allows users to upload images
  - A server application which accepts SOAP requests
  - A server application with an XML-based REST API

JPEG 2000 image exploit on the OpenJPEG openjp2 version 2.1.1
A specially crafted image caused a buffer overflow in the JPEG code and overwrites memory in such a way to allow allow arbitrary code execution

https://www.flickr.com/photos/mediumpanda/

https://www.flickr.com/photos/jasonahowie/

https://dzone.com/articles/will-it-pwn-cve-2017-563 emote-code-execution-in

```
Content-Type: %{(#_='multipar
(#_memberAccess=@ognl.Ognl                    R_ACCESS)
.
(@java.lang.Runtime@g
localhost:8000'))}
```

OGNL

Equifax

uts

If type contains   m-data'

try to parse

This fa   part of the building an error message the OGNL is evaluated…

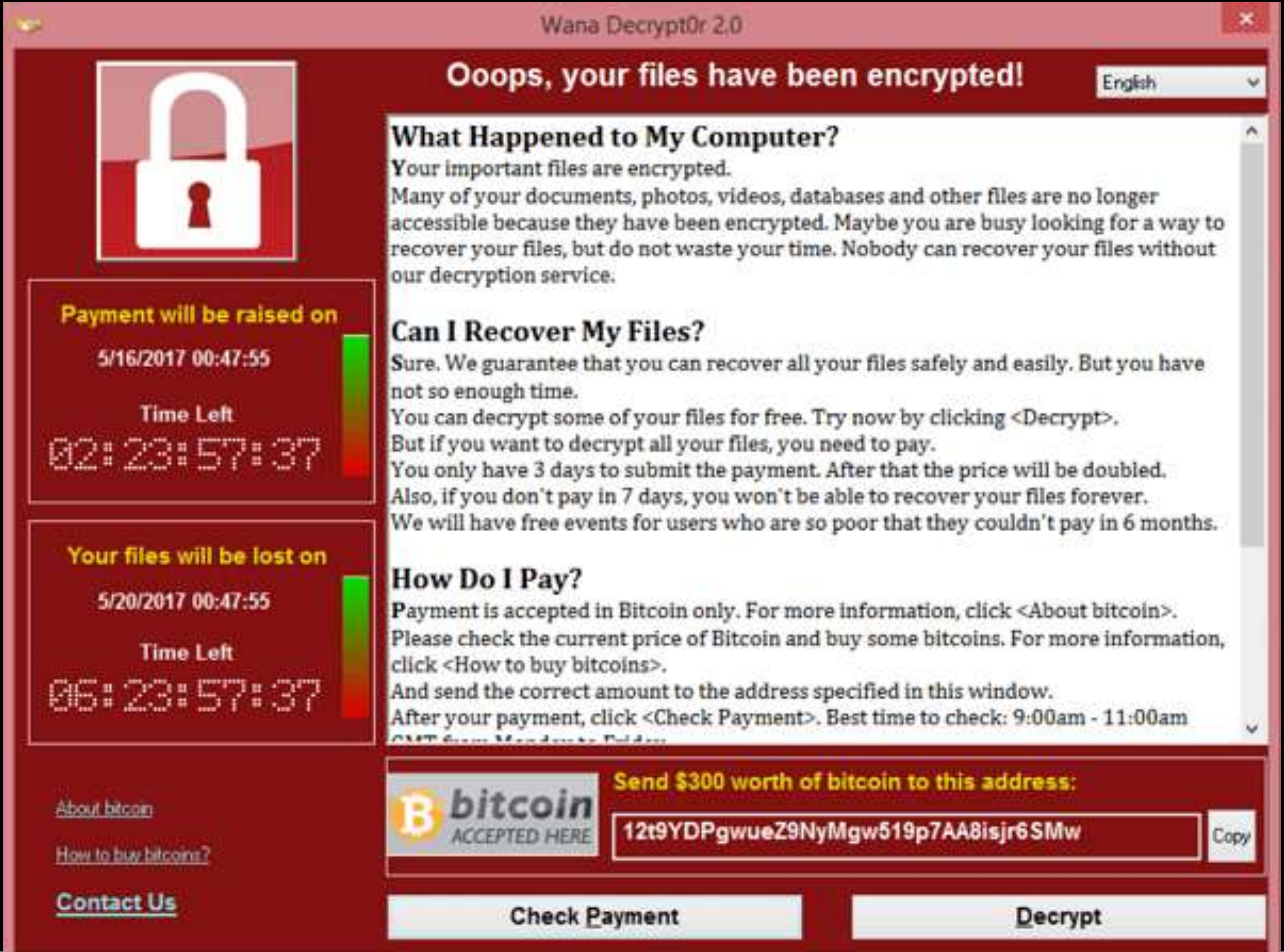
Attack Vectors – Cryptographic Issues

- Protocol flaws
  - Many implementations affected (Java, GSKit, OpenSSL, Browsers)
  - BEAST, POODLE, SLOTH, Logjam, Bar Mitzvah
- Implementation flaws
  - Specific to Java
  - No fancy names!
- Variable Impact
  - DoS
  - Private key exposure
  - Decryption of encrypted data

Attack Vectors – Local

- Can only be exploited by a local user
- Exposure of sensitive data
  - E.g. temp files with inappropriate permissions
- Code injection
  - E.g. native code loaded from an unexpected location due to bad LIBPATH
- May lead to a remotely exploitable vulnerability
  - E.g. local user plants malicious code which can be executed remotely
- Usually low CVSS due to the access required

You don’t think you’re vulnerable to local attacks?



Ooops, your files have been encrypted!

What Happened to My Computer?

Your important files are encrypted.
Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.

Can I Recover My Files?

Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.
You can decrypt some of your files for free. Try now by clicking <Decrypt>.
But if you want to decrypt all your files, you need to pay.
You only have 3 days to submit the payment. After that the price will be doubled.
Also, if you don't pay in 7 days, you won't be able to recover your files forever.
We will have free events for users who are so poor that they couldn't pay in 6 months.

How Do I Pay?

Payment is accepted in Bitcoin only. For more information, click <About bitcoin>.
Please check the current price of Bitcoin and buy some bitcoins. For more information, click <How to buy bitcoins>.
And send the correct amount to the address specified in this window.
After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am

Payment will be raised on
5/16/2017 00:47:55
Time Left
02:23:57:37

Your files will be lost on
5/20/2017 00:47:55
Time Left
06:23:57:37

Send $300 worth of bitcoin to this address:
12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw

https://en.wikipedia.org/w/index.php?curid=54032765

WANNA CRY?

Steve Poole

Wanna Cry

https://en.wikipedia.org/wiki/WannaCry_ransomware_attack

UK: National Health Service impacted:

India: All ATMs closed

Nissan: Halted all production

Renault: Halted some production

- Friday, 12 May 2017
- Has infected 250K computers in 150+ countries
- It encrypts data and holds it for ransom
- The computer owner has a limited time to pay (in bitcoin) about $500

- So far the bitcoin owners have received about 50 bitcoins ~= $85K ($3/infected machine)

Your Java server is not as secure as you think

More Examples

Deserialization Vulnerabilities

- Abuse of the readObject() method of one or more classes
  - readObject() is invoked **before** the data is deserialized
- Attackers use “gadget chains” in classes on the server’s classpath
  - Usually a complex set of nested objects which result in a Runtime.exec()
  - Costly to fix – servers may have multiple copies of the vulnerable code
- **Applications are vulnerable if they:**
  - **Have the vulnerable code on their classpath**
  - **Accept untrusted serialized data (this is much more common that you might think!)**
- Known for years, but came to prominence with problem in Apache Commons remote code execution
  - Many products affected – e.g. WebLogic, WebSphere, Tomcat, JBoss, Jenkins
- Serialization filtering was added to the Java runtime in January 2017
  - JEP 290 – Filter Incoming Serialization Data
  - Allows classes to be whitelisted

Example vulnerability – JDWP

- JDWP = Java Debug Wire Protocol
    - Disabled by default – enabled with a command line option
    - JVM internals can be observed and modified remotely
    - No authentication required
    - Classes can be changed, code can be injected
- Well-known online banking website
    - **JDWP enabled on public facing servers**
    - Presumably JDWP was enabled in development/test…
- Found by a researcher after a simple port scan
    - Bug bounty => ££££
- Simple fix – disable JDWP!

More Code

@spoole167

```
public class HelpfulClassLoader {

  private Properties p=new Properties(System.getProperties());

  public HelpfulClassLoader() {

      p.put("default", ”foo.StringHandler");
      p.put("foo",     "com.ibm.runtimes.demo.foo");

  }

  public Class loadClassHelpfully(String handler) throws ClassNotFoundException {

    String className=p.getProperty(handler);
    try {
              return Class.forName(className);
     } catch (Exception e) {
              throw new ClassNotFoundException("could not create class for handler”                    +handler+" with
value "+className);

}}}
```

handler	Class name
default	foo.StringHandler
foo	com.ibm.runtimes.demo.foo

```
HelpfulClassLoader h=new HelpfulClassLoader();

try {
            Class c=h.loadClassHelpfully("default");

      System.out.println("class for default="+c);

} catch (ClassNotFoundException e) {

            e.printStackTrace();

}
```

handler	Class name
default	foo.StringHandler
foo	com.ibm.runtimes.demo.foo

```
HelpfulClassLoader h=new HelpfulClassLoader();

try {
            Class c=h.loadClassHelpfully("default");

      System.out.println("class for default="+c);

} catch (ClassNotFoundException e) {

            e.printStackTrace();

}
```

```
class for default=class foo.StringHandler
```

handler	Class name
default	foo.StringHandler
foo	com.ibm.runtimes.demo.foo

```
HelpfulClassLoader h=new HelpfulClassLoader();

try {
        Class c=h.loadClassHelpfully(”foo");

    System.out.println("class for foo="+c);

} catch (ClassNotFoundException e) {

        e.printStackTrace();

}
```

handler	Class name
default	foo.StringHandler
foo	com.ibm.runtimes.demo.foo

```
HelpfulClassLoader h=new HelpfulClassLoader();

try {
        Class c=h.loadClassHelpfully("foo");

    System.out.println("class for foo="+c);

} catch (ClassNotFoundException e) {

        e.printStackTrace();

}
```

```
java.lang.ClassNotFoundException: could not create class for handler foo with value com.ibm.runtimes.demo.foo
        at
com.ibm.runtimes.demo.vulnerabilities.samples.HelpfulClassLoader.loadClassHelpfully(HelpfulClassLoader.java:23)
        at com.ibm.runtimes.demo.vulnerabilities.samples.HelpfulClassLoader.main(HelpfulClassLoader.java:40)
```

handler	Class name
default	foo.StringHandler
foo	com.ibm.runtimes.demo.foo

```
HelpfulClassLoader h=new HelpfulClassLoader();

try {

        Class c=h.loadClassHelpfully(”java.ext.dirs");

} catch (ClassNotFoundException e) {

        e.printStackTrace();

}
```

handler	Class name
default	foo.StringHandler
foo	com.ibm.runtimes.demo.foo

```
HelpfulClassLoader h=new HelpfulClassLoader();

try {

        Class c=h.loadClassHelpfully("java.home");

} catch (ClassNotFoundException e) {

        e.printStackTrace();

}
```

```
java.lang.ClassNotFoundException: could not create class for handler java.home with value
/Users/spoole/Library/Java/Extensions:
/Library/Java/JavaVirtualMachines/jdk1.8.0_102.jdk/Contents/Home/jre/lib/ext:
/Library/Java/Extensions:/Network/Library/Java/Extensions:
/System/Library/Java/Extensions:
/usr/lib/java
```

Something like this helpful code

Coupled with the missing URL decoder check and the remote execution code inside Wanna Cry

And your Java application is compromised.

“A vulnerability is a bug which can be exploited by an attacker”

“A vulnerability is also a feature which can be exploited by an attacker”

“A vulnerability is a bug which can be exploited by an attacker”

“A vulnerability is also a feature which can be exploited by an attacker”

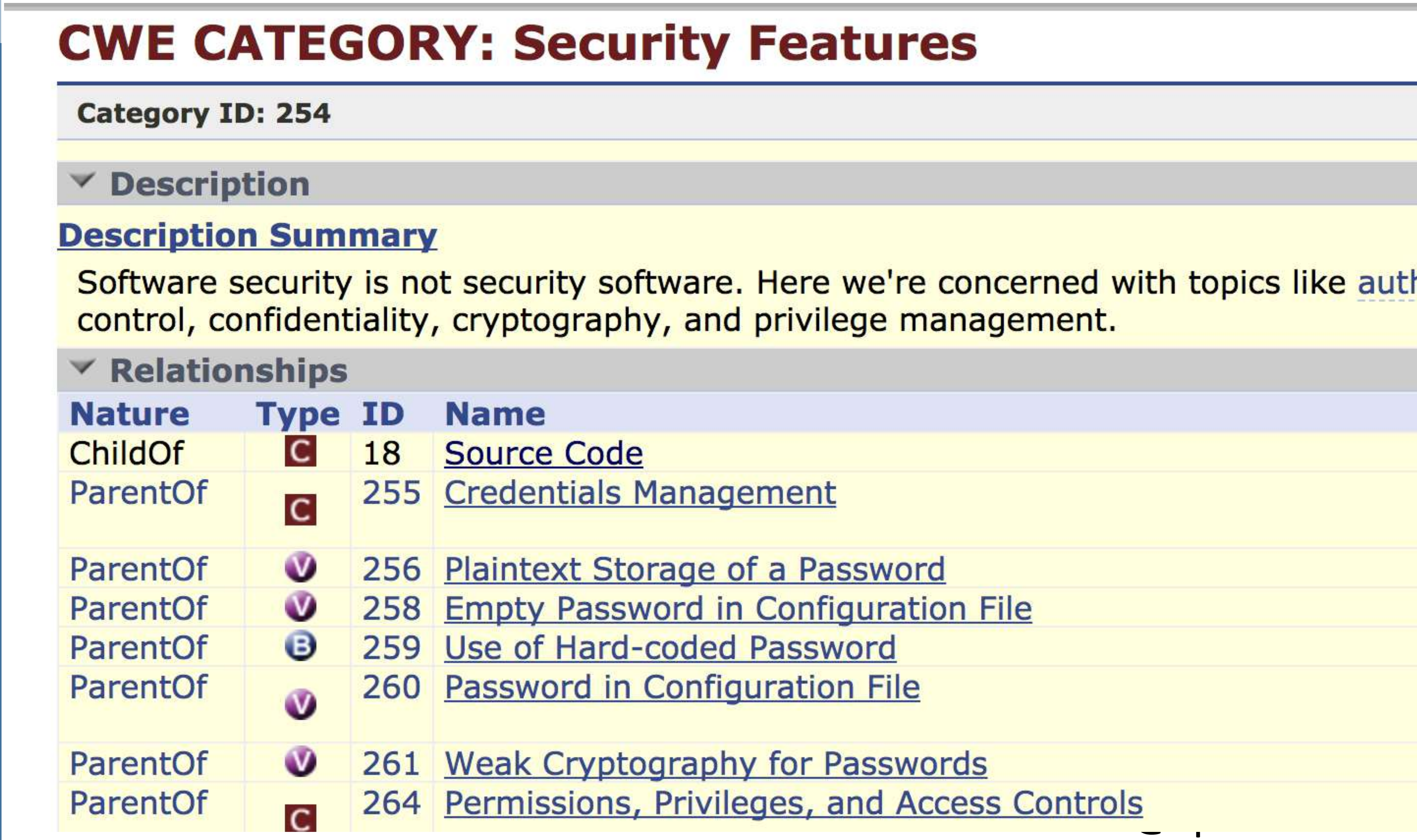
“A vulnerability is also a developer aid which can be exploited by an attacker”

Helping you to be more informed

@spoole167

Coding Practises

cwe.mitre.org



CWE CATEGORY: Security Features

Category ID: 254

Description

Description Summary

Software security is not security software. Here we're concerned with topics like auth
control, confidentiality, cryptography, and privilege management.

Relationships

Nature	Type	ID	Name
ChildOf	C	18	Source Code
ParentOf	C	255	Credentials Management
ParentOf	V	256	Plaintext Storage of a Password
ParentOf	V	258	Empty Password in Configuration File
ParentOf	B	259	Use of Hard-coded Password
ParentOf	V	260	Password in Configuration File
ParentOf	V	261	Weak Cryptography for Passwords
ParentOf	C	264	Permissions, Privileges, and Access Controls

Example 2

The following code is an example of an internal hard-coded password in the back-end:

Example Languages: **C and C++** *(Bad Code)*

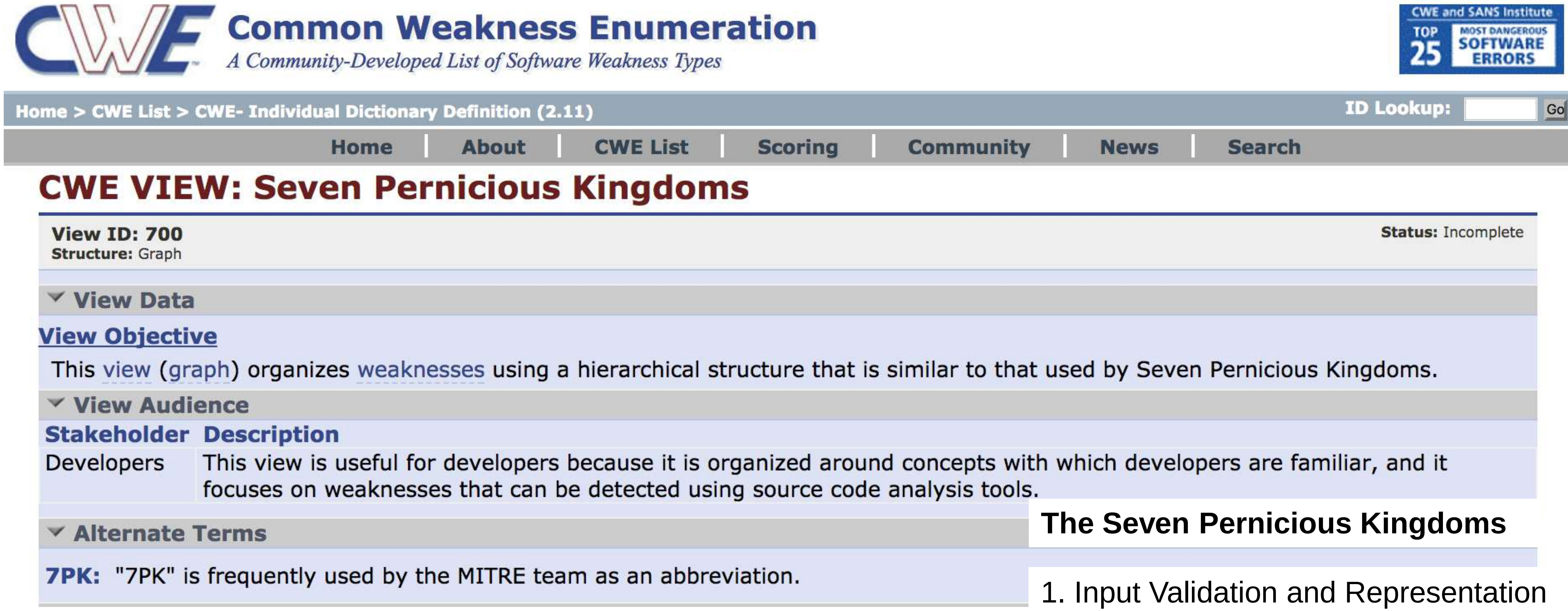
```
int VerifyAdmin(char *password) {
  if (strcmp(password, "Mew!")) {

    printf("Incorrect Password!\n");
    return(0)
  }
  printf("Entering Diagnostic Mode...\n");
  return(1);
}
```

Example Language: **Java** *(Bad Code)*

```
int VerifyAdmin(String password) {
  if (!password.equals("Mew!")) {
    return(0)
  }
  //Diagnostic Mode
  return(1);
}
```

Every instance of this program can be placed into diagnostic mode with the same password. Even worse is the fact that if this program is distributed as a binary-only distribution, it is very difficult to change that password or disable this "functionality."


Common Weakness Enumeration
A Community-Developed List of Software Weakness Types
CWE and SANS Institute TOP 25 MOST DANGEROUS SOFTWARE ERRORS
Home > CWE List > CWE- Individual Dictionary Definition (2.11)
ID Lookup:
Go
Home | About | CWE List | Scoring | Community | News | Search

CWE VIEW: Seven Pernicious Kingdoms

View ID: 700
Structure: Graph

Status: Incomplete

View Data

View Objective

This view (graph) organizes weaknesses using a hierarchical structure that is similar to that used by Seven Pernicious Kingdoms.

View Audience

Stakeholder	Description
Developers	This view is useful for developers because it is organized around concepts with which developers are familiar, and it focuses on weaknesses that can be detected using source code analysis tools.

Alternate Terms

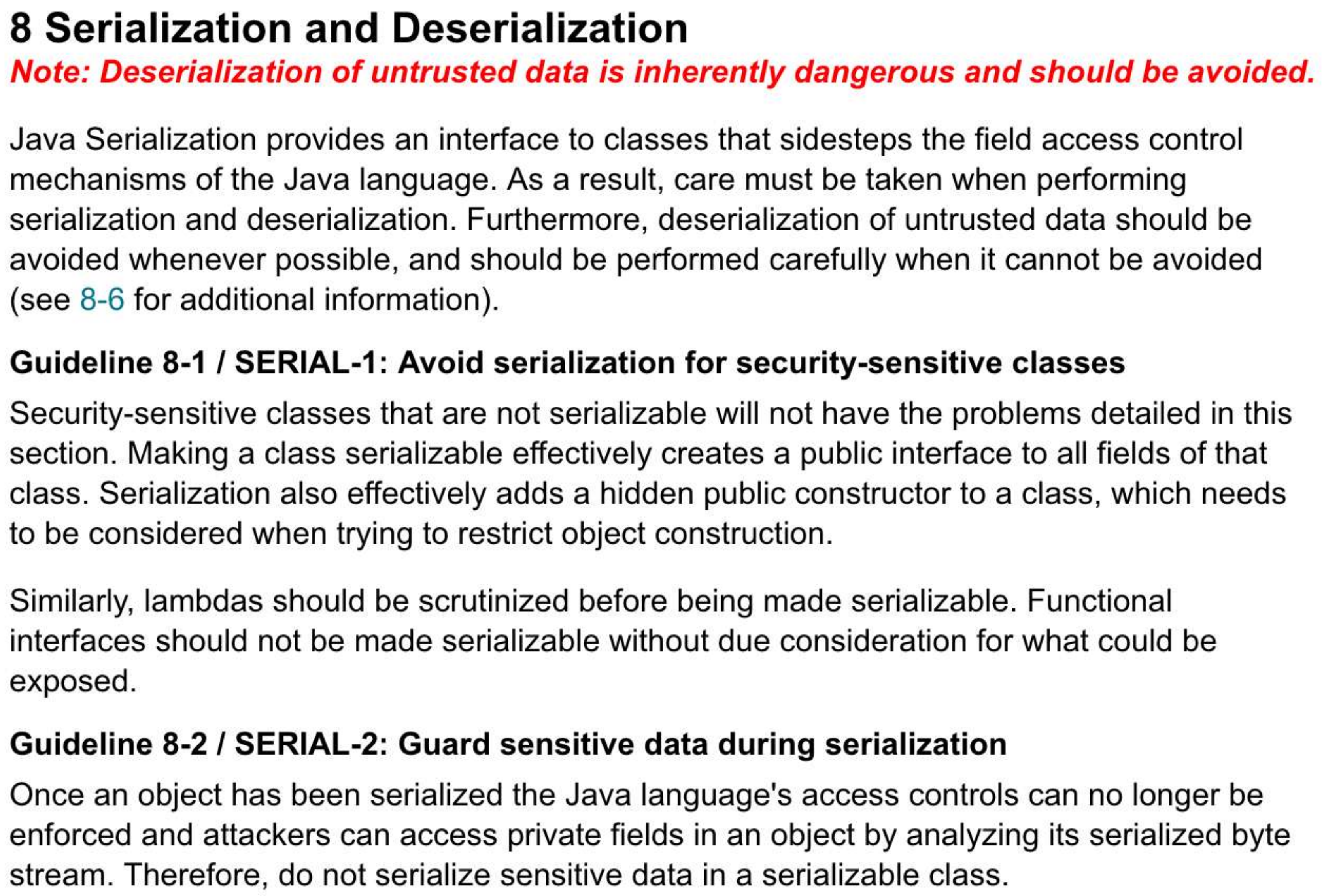
7PK: "7PK" is frequently used by the MITRE team as an abbreviation.

The Seven Pernicious Kingdoms

1. Input Validation and Representation
2. API Abuse
3. Security Features
4. Time and State
5. Error Handling
6. Code Quality
7. Encapsulation

* Environment

Secure Coding Guidelines for Java SE



8 Serialization and Deserialization

Note: Deserialization of untrusted data is inherently dangerous and should be avoided.

Java Serialization provides an interface to classes that sidesteps the field access control mechanisms of the Java language. As a result, care must be taken when performing serialization and deserialization. Furthermore, deserialization of untrusted data should be avoided whenever possible, and should be performed carefully when it cannot be avoided (see 8-6 for additional information).

Guideline 8-1 / SERIAL-1: Avoid serialization for security-sensitive classes

Security-sensitive classes that are not serializable will not have the problems detailed in this section. Making a class serializable effectively creates a public interface to all fields of that class. Serialization also effectively adds a hidden public constructor to a class, which needs to be considered when trying to restrict object construction.

Similarly, lambdas should be scrutinized before being made serializable. Functional interfaces should not be made serializable without due consideration for what could be exposed.

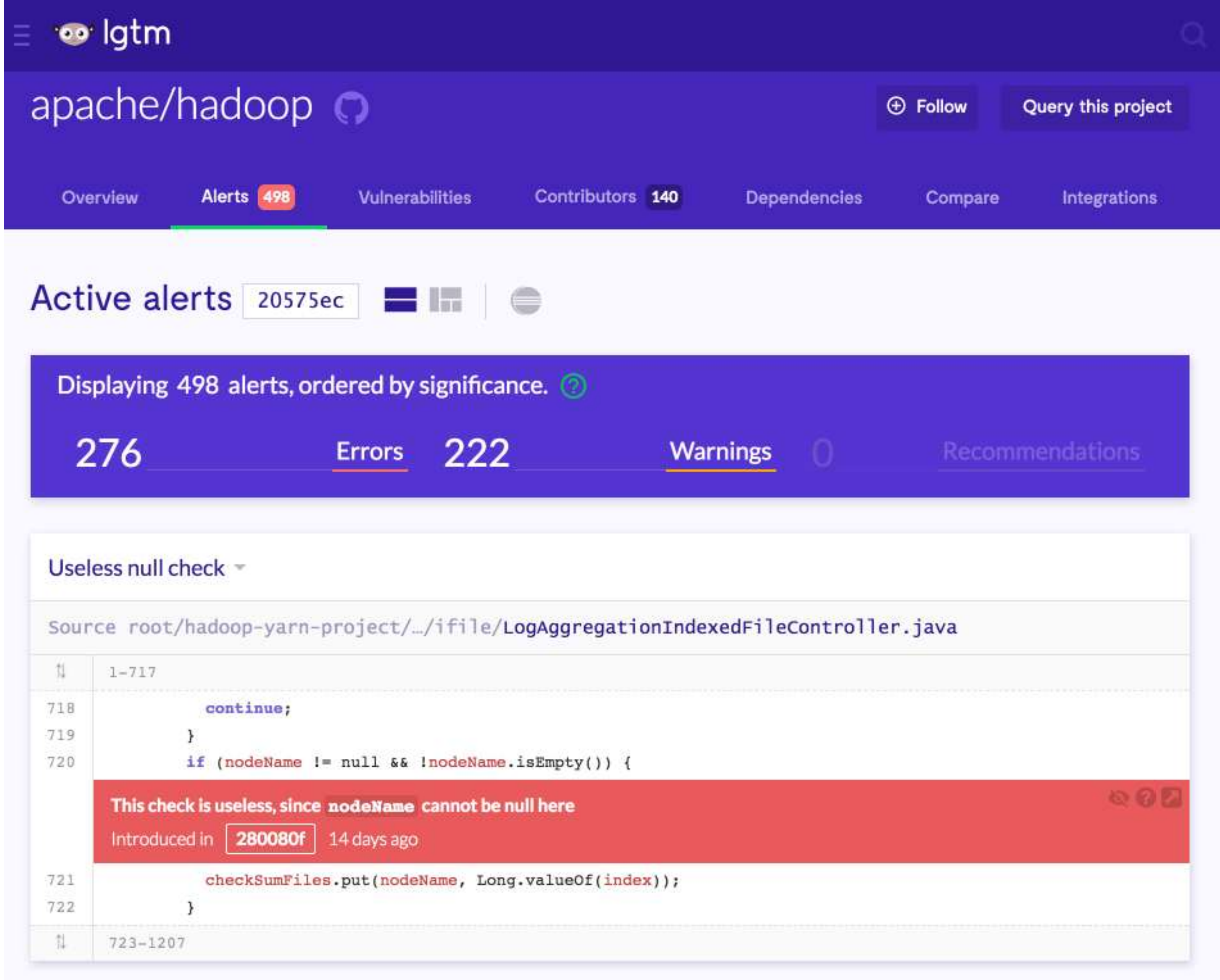
Guideline 8-2 / SERIAL-2: Guard sensitive data during serialization

Once an object has been serialized the Java language's access controls can no longer be enforced and attackers can access private fields in an object by analyzing its serialized byte stream. Therefore, do not serialize sensitive data in a serializable class.

http://www.oracle.com/technetwork/java/seccodeguide-139067.html

@spoole167

Analysis Tools



Displaying 498 alerts, ordered by significance.

276 Errors 222 Warnings 0 Recommendations

Useless null check

Source root/hadoop-yarn-project/…/ifile/LogAggregationIndexedFileController.java

```
1-717
                continue;
            }
            if (nodeName != null && !nodeName.isEmpty()) {
```

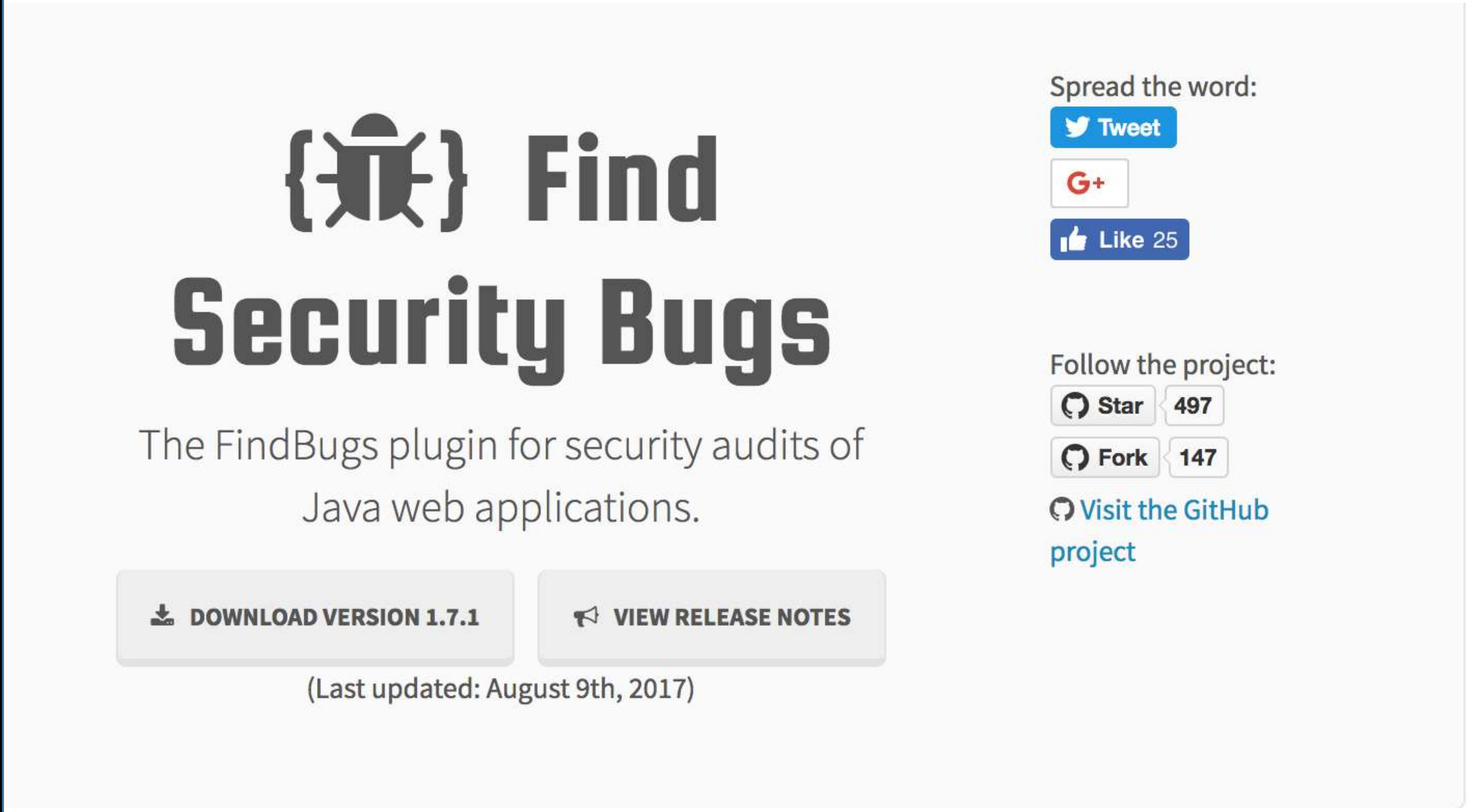
This check is useless, since `nodeName` cannot be null here

Introduced in 280080f 14 days ago

```
                checkSumFiles.put(nodeName, Long.valueOf(index));
            }
723-1207
```

Analysis Tools

find-sec-bugs.github.io



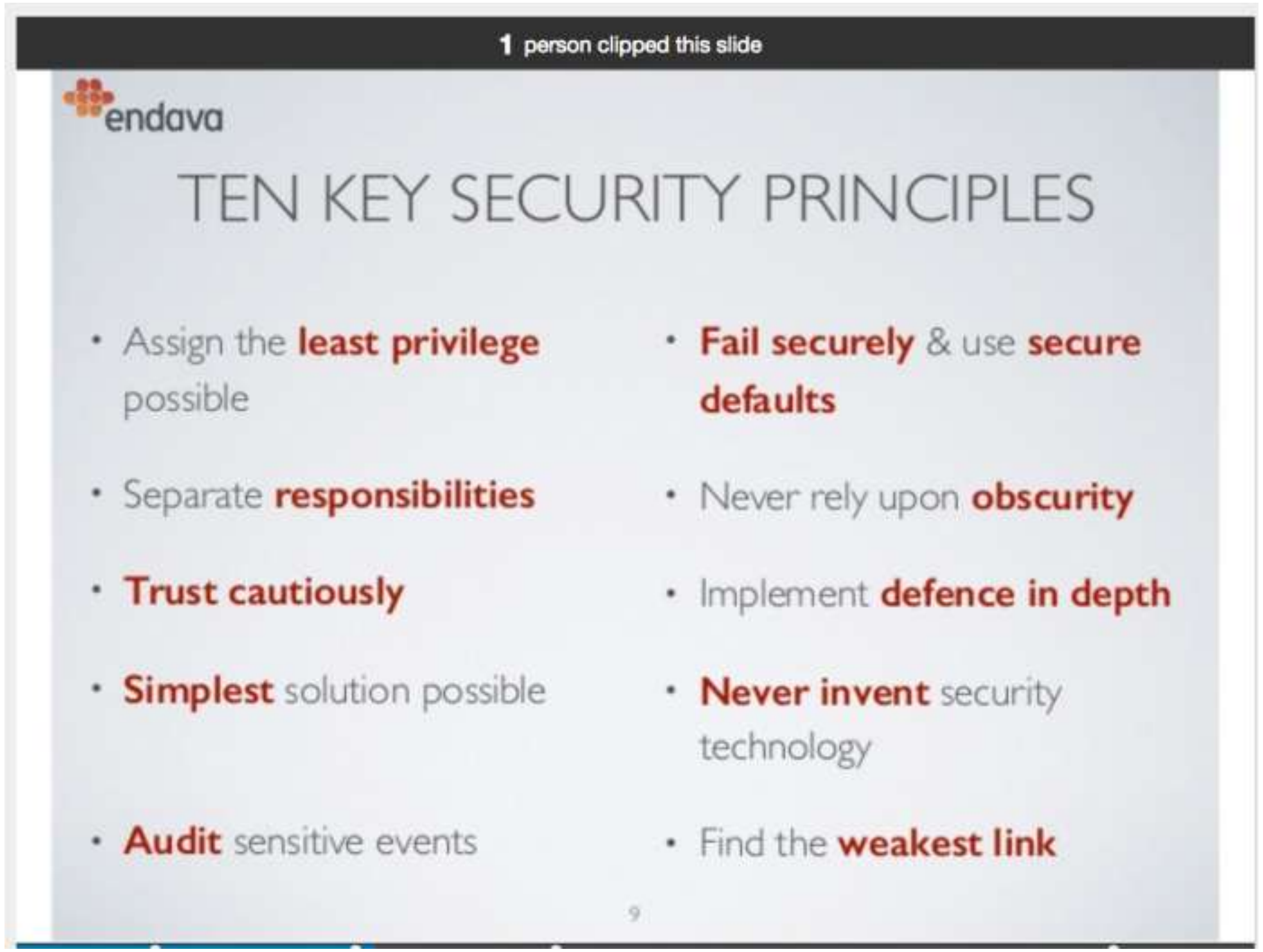
Find Security Bugs

The FindBugs plugin for security audits of Java web applications.

(Last updated: August 9th, 2017)

Online Guides

Secure by Design - Security Design Principles for the Rest of Us


1 person clipped this slide

endava

TEN KEY SECURITY PRINCIPLES

- Assign the **least privilege** possible
- Separate **responsibilities**
- **Trust cautiously**
- **Simplest** solution possible
- **Audit** sensitive events
- **Fail securely** & use **secure defaults**
- Never rely upon **obscurity**
- Implement **defence in depth**
- **Never invent** security technology
- Find the **weakest link**

9

https://www.slideshare.net/EoinWoods1/secure-by-design-security-design-principles-for-the-rest-of-us

Summary

@spoole167

Reducing Risk

- Keep all Java instances up to date
- Use vulnerability scanning tools
  - Various around. From source code level to binary signature analysis.
- Don’t write custom security managers
- Don’t write custom crypto code
- Don’t write custom XML parsers
- Use modern encryption protocols and algorithms
- Be very careful with:
  - Untrusted data , Deserialization (including RMI and JMX)
  - JDWP, Runtime.exec()
  - Native code

If you are receiving data of any sort: validate it.

Keeping safe: it’s in your hands

Keep current. Every vulnerability fix you apply is one less way in.
Compartmentalize. Separate data, code, access controls etc.
Just like bulkhead doors in a ship: ensure one compromise doesn’t sink your boat.
Design for intrusion. Review you levels of ‘helpfulness’ and flexibility
Learn about Penetration Testing
Learn about security tools & services - IBM App Scan, lgtm.com
Learn about secure coding - https://cwe.mitre.org
http://www.oracle.com/technetwork/java/seccodeguide-139067.html ,

Understand that making your development life easier makes the hackers job easier

There are bad guys out there and your application is at risk

Don’t make it worse by ignoring the problem

https://www.flickr.com/photos/koolmann/

Thank you

Any questions?

https://www.flickr.com/photos/koolmann/