



**MANICODE**  
SECURE CODING EDUCATION

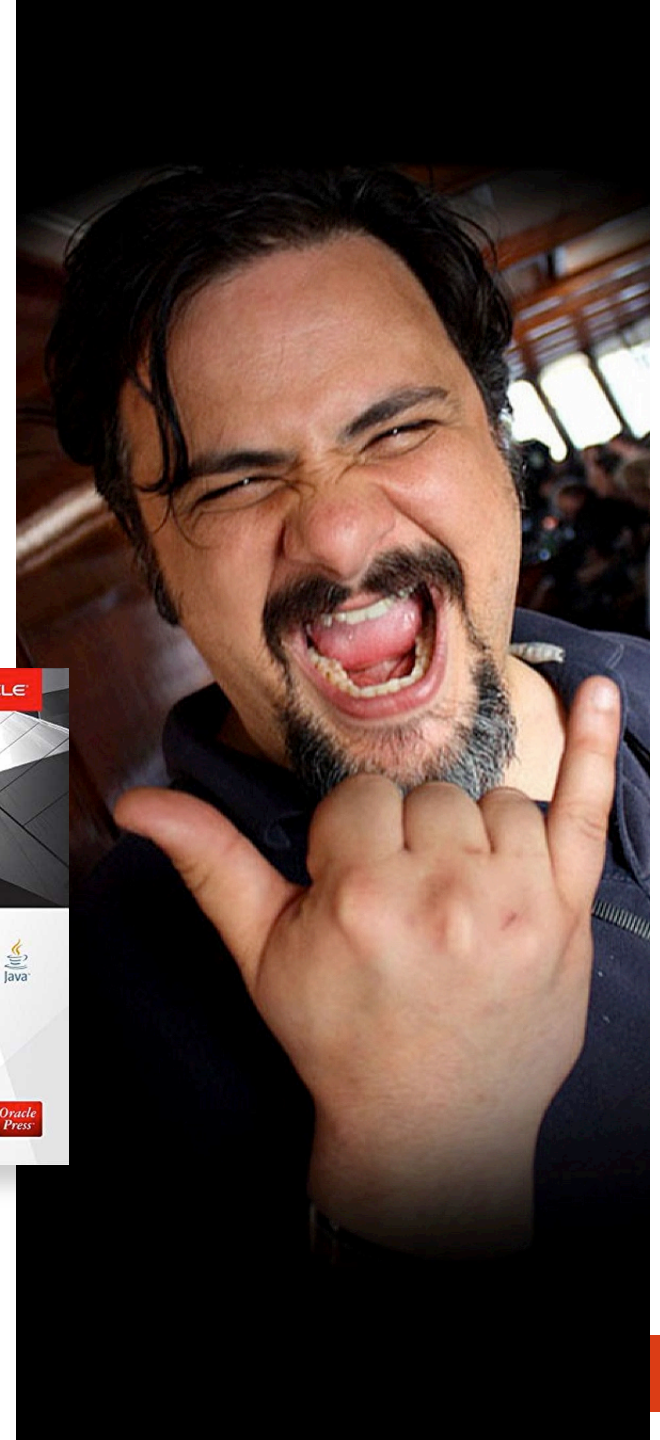
# Java 9 Security Enhancements

# A little background dirt...

[jim@manicode.com](mailto:jim@manicode.com)

 [@manicode](https://twitter.com/manicode)

- Former OWASP Global Board Member
- Project manager of the OWASP Cheat Sheet Series and several other OWASP projects
- 20+ years of software development experience
- Author of "Iron-Clad Java, Building Secure Web Applications" from McGraw-Hill/Oracle-Press
- Kauai, Hawaii Resident



# Java 9 Security JEP's

# Java 9 Security Enhancements

- There are a *plethora* of security related JEPs for JDK 9:

**219: Datagram Transport Layer Security (DTLS)**

229: Create PKCS12 Keystores by Default

232: Improve Secure Application Performance

**244: TLS ALPN Extension**

246: Leverage CPU Instructions for GHASH and RSA

**249: OCSP Stapling for TLS**

**273: DRBG-Based SecureRandom Implementations**

287: Support **SHA-3** Hash Algorithms

**288: Disable SHA-1 Certificates**

**290: Filter Incoming Serialization Data**



# JEP 219

# DTLS

# JEP 219

## Datagram Transport Layer Security (DTLS)

<http://openjdk.java.net/jeps/219>

- **TLS is not sufficient** for a variety of datagram apps
- Datagram applications still **need transport security!**
- JEP 219 defines an API for Datagram Transport Layer Security (DTLS) version 1.0 (**RFC 4347**) and 1.2 (**RFC 6347**)
- Concise implementation that is **transport-independent** and similar to `javax.net.ssl.SSLEngine`

# JEP 244 ALPN

# JEP 244

## TLS *Application-Layer Protocol Negotiation Extension*

<http://openjdk.java.net/jeps/244>

- JEP 244 extends javax.net.ssl package to support the TLS Application Layer Protocol Negotiation (ALPN) Extension
- When different protocols are supported on the same TCP or UDP port, ALPN allows a negotiation to determine **which protocol will be used with a TLS connection**
- An important consumer of this enhancement is the **HTTP/2 client** (JEP 110)

# JEP 249

# OCSP Stapling

# JEP 249

## OCSP Stapling for TLS

<http://openjdk.java.net/jeps/249>

- JEP 249 implements **OCSP Stapling for TLS clients**
- Revocation (in general) does not work well

It takes at least **10 days** for the revocation Information to fully *propagate*

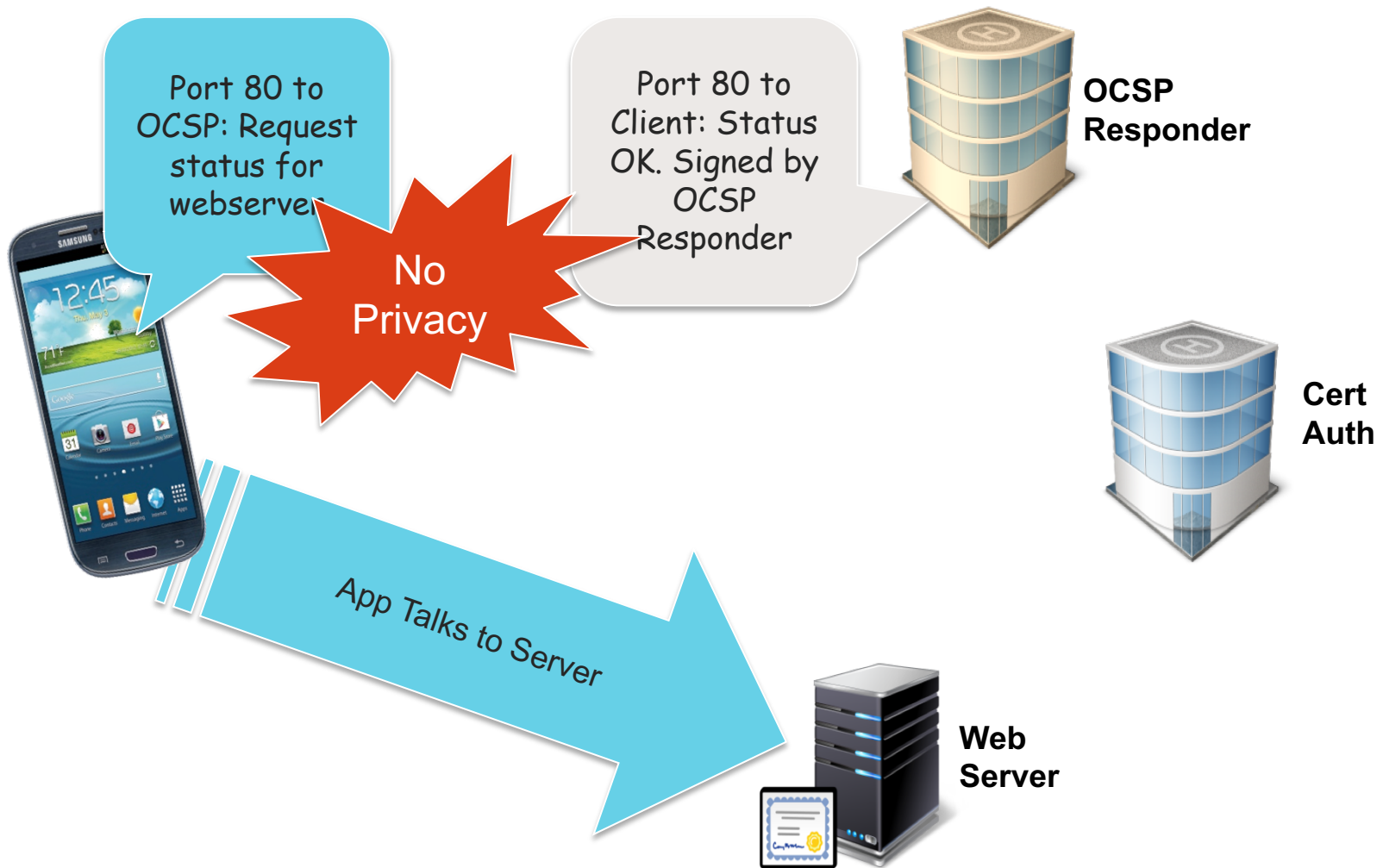
Browser *soft fail policy* makes revocation *ineffective*

Some *OCSP* request can be *intercepted*

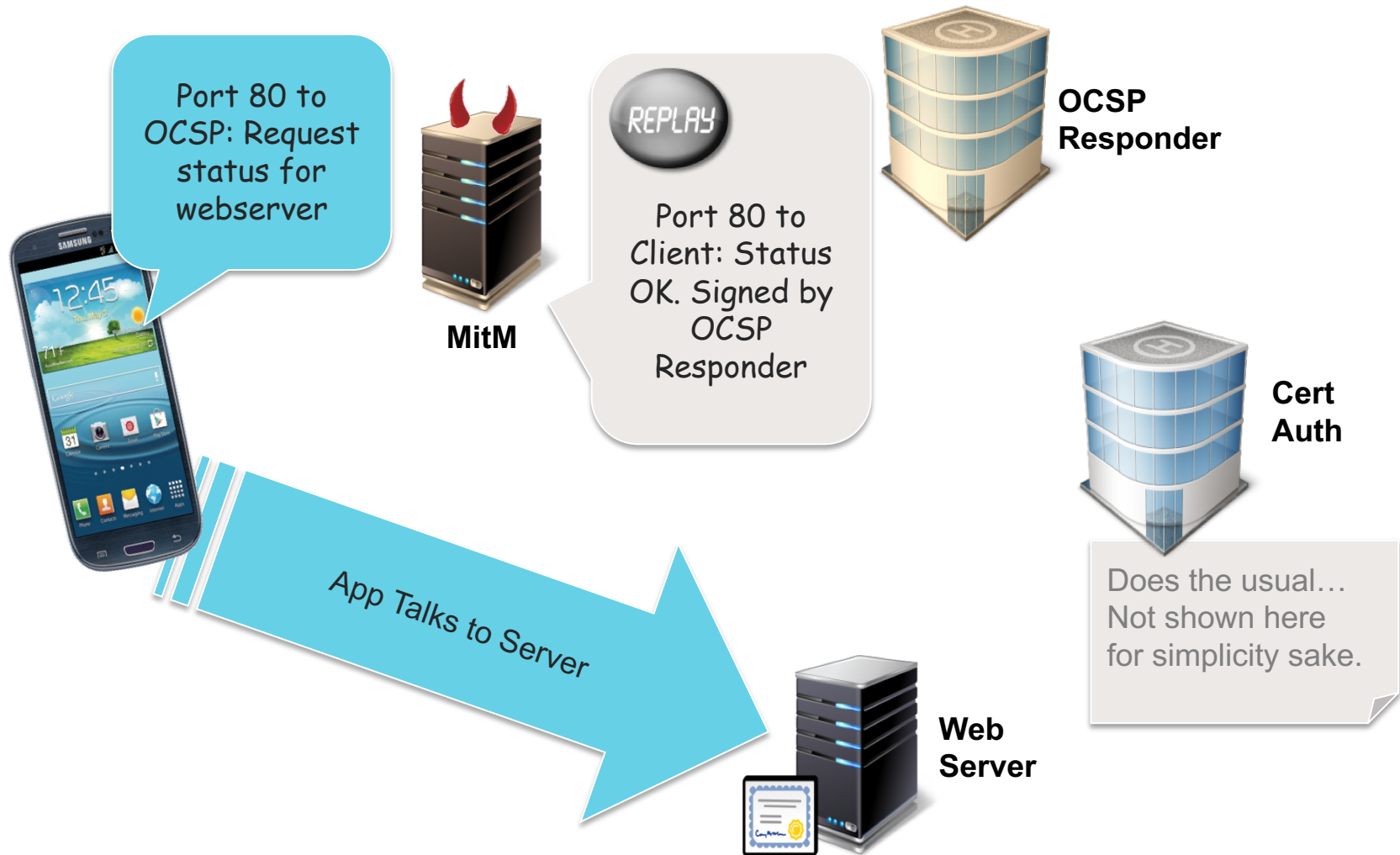
**Most browsers ignore revocation** for all certificates but EV certificates

# Online Certificate Status Protocol (OCSP)

An alternative to certificate lists (CRL)

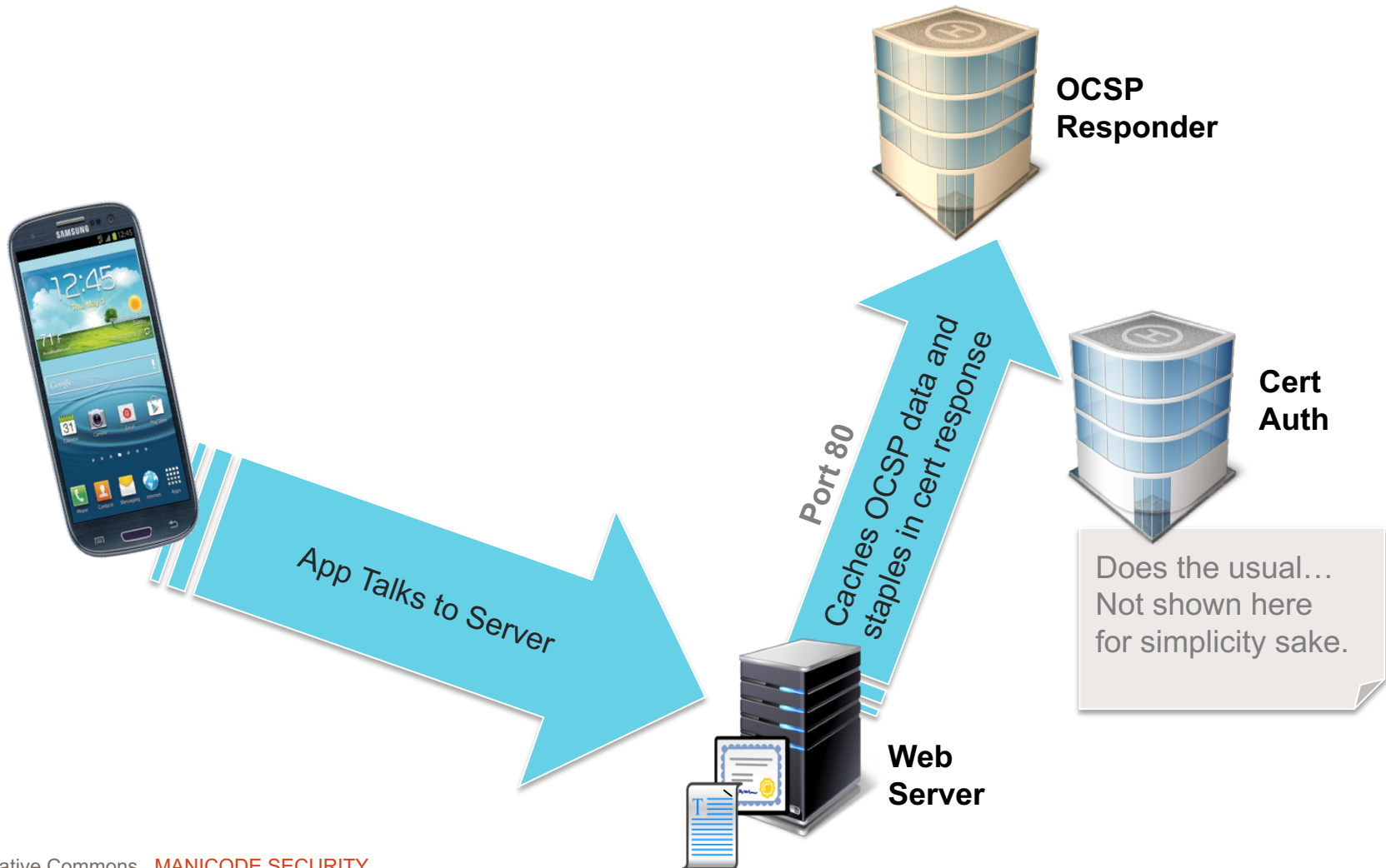


# Attacks against OCSP





# OCSP Stapling Faster, safer and more private



# JEP 273

# DRBG

# JEP 273

## Deterministic Random Bit Generator (DRBG)

<http://openjdk.java.net/jeps/273>

- NIST 800-90Ar1 defines three DRBGs: **Hash\_DRBG**, **HMAC\_DRBG** and **CTR\_DRBG**.
- These new DRBG's use strong modern algorithms such as SHA-512 and AES-256.
- **API changes**
  - New SecureRandom methods specifying additional input in the course of **seeding, reseeding, and random-bit generation**.
  - New methods in SecureRandomSpi, to implement the new methods above.
  - A new SecureRandomParameters interface so that additional input can be provided to the new SecureRandom methods.
  - A new DrbgParameters class (and its inner classes) implementing SecureRandomParameters to be used by DRBG.

# JEP 287

# SHA-3

# JEP 287

## SHA-3 Implementation

<http://openjdk.java.net/jeps/287>

- FIPS 202 defines four new hash functions: **SHA3-224, SHA3-256, SHA3-384, and SHA3-512**. These can be implemented as new algorithms of the java.security.MessageDigest API under the standard names "SHA3-224", "SHA3-256", "SHA3-384", and "SHA3-512".
- **No new APIs are necessary**, since there are no new parameters required.
- Here is the **list of providers and the corresponding algorithm** enhancements:
  - "SUN" provider: SHA3-224, SHA3-256, SHA3-384, and SHA3-512
  - "OracleUcrypto" provider: SHA-3 digests supported by Solaris 12.0

# JEP 288

## Disable SHA-1 Certificates

# TLS Benefits

## Confidentiality

Spy cannot view your data

## Integrity

Spy cannot change your data

## Authenticity

*Server you are visiting is the right one,  
backed up by the Certificate Authority System*



# TLS Certificates



TLS uses X.509 Certificates

Used to authenticate the other party

NOT used to help negotiate a symmetric key (beyond authentication)

TLS certificates from certificate authorities help websites prove their authenticity. These certificates contain:

- The certificate holder
- The domain that the certificate was issued to
- ***The signature of the Certificate Authority who verified the certificate***



# JEP 288

## Disable SHA-1 Certificates

<http://openjdk.java.net/jeps/288>


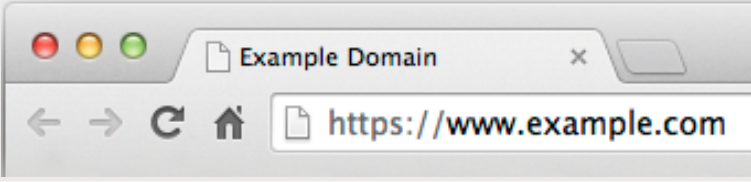
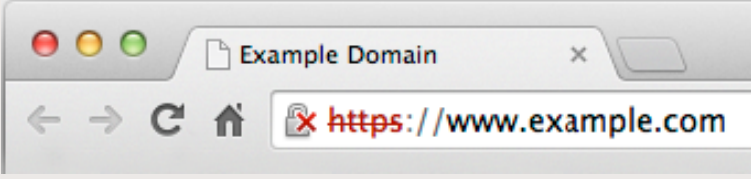
- **Disable** X.509 certificate chains with **SHA-1 based digital signatures**.
- *"SHA-1 should no longer be used to apply digital signatures to data"*
  - <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf>
  - NIST Recommendation for Key Management, Part 1: General
- *"CAs MAY continue to sign certificates to verify OCSP responses using SHA1 until January 1, 2017"*
  - v1.3 of the CA/Browser Forum Baseline Requirements (Section 7.1.3)
  - <https://cabforum.org/documents/#Baseline-Requirements>

# SHA-1 and the Urgency to Move On

- In 2005, cryptographers proved that SHA-1 could be **cracked 2,000 times** faster than predicted.
- At one point **90% of websites** used SHA-1 to protect themselves from being impersonated. That number is now below 4%.
- **As long as browsers need to support SHA-1** for someone, anyone's **certificate can be forged** because browsers will not know there is a good cert that uses something better

Year	Cost (In US\$)	Cost Within Reach For
2012	2,770,000	Government, large corporations
2016	700,000	Medium size institutions
2018	173,000	Organized crime
2021	43,000	University research

# The Death of SHA-1 according to Google

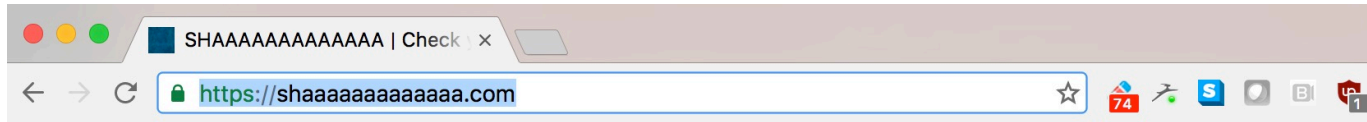
Chrome Version	Date	UI Changes	Behavior
39	Sept 2014		Certs that expire in Jan 2017 using SHA-1 or mixed content
40	Nov 2014		Certs that expire between 1 June 2016 to 31 Dec 2016 using SHA-1 in the chain
41	Q1 2016		Certs that expire on or after 1 Jan 2017

Source: <http://googleonlinesecurity.blogspot.com/2014/09/gradually-sunseting-sha-1.html>

## The Death of SHA-1 according to Mozilla

- Show the “Untrusted Connection” error **whenever a publically issued SHA-1 certificate issued after January 1, 2016**, is encountered in Firefox.
  - Locally installed authorities (like MITM proxy tools) are NOT subject to this rule.
- Firefox will show the "Untrusted Connection" error message for all **SHA-1-based certificates after January 2017**.
- <https://www.fxsitecompat.com/en-CA/docs/2015/sha-1-based-certificates-with-validity-period-from-2016-will-not-be-validated/>

<https://shaaaaaaaaaaaaa.com/>



# SHAAAAAAAAAAAAA

Check your site for weak SHA-1 certificates. [Open source](#), by [@konklone](#).

Check above to see if a site is still using certificates that were issued using the [dangerously outdated](#) SHA-1 signature algorithm.

As of **January 1, 2016**, no publicly trusted CA is allowed to issue a SHA-1 certificate. So any new certificate you get should automatically use a SHA-2 algorithm for its signature.

However, existing SHA-1 certificates are still trusted by modern browsers and operating systems. Generally, they will be removing support for SHA-1 entirely by January 1, 2017.

Legacy clients will continue to accept SHA-1 certificates, and it is possible to have requested a certificate on December 31, 2015 valid for 39 months. So, it is possible to see SHA-1 certificates in the wild that expire in 2019.

## Credits

This website is an [open source](#) project, and includes a [command line tool](#) — please [lend a hand!](#)

Thanks to [Mathias Bynens](#), [Justin Mayer](#), and [Jonny Barnes](#) for inspiration and assistance.

# SHA1 Collisions No Longer Theoretical

## February, 2017



## Google Security Blog

The latest news and insights from Google on security and safety on the Internet

### Announcing the first SHA1 collision

February 23, 2017

Posted by Marc Stevens (CWI Amsterdam), Elie Bursztein (Google), Pierre Karpman (CWI Amsterdam), Ange Albertini (Google), Yarik Markov (Google), Alex Petit Bianco (Google), Clement Baisse (Google)

Cryptographic hash functions like SHA-1 are a cryptographer's swiss army knife. You'll find that hashes play a role in browser security, managing code repositories, or even just detecting duplicate files in storage. Hash functions compress large amounts of data into a small message digest. As a cryptographic requirement for wide-spread use, finding two messages that lead to the same digest should be computationally

# Disable SHA-1 TLS Server Certificates in Java

## July 18, 2017

**Disabled SHA-1 in TLS Server certificate chains anchored by roots included by default in Oracle's JDK; local or enterprise CAs are not effected**

2017-07-18 Released Java 9, 8u141 b15, 7u151 b15, 6u151 b15, R28.3.15

- 2017-05-02 Target date changed from 2017-10-17 to 2017-07-18.
- 2017-03-06 Target date changed from 2017-04-18 to 2017-10-17. Narrowed scope from all SHA-1 usage: only TLS will be affected, code signing will not be affected at this time.
- 2016-08-18 Announced



# Crypto Note

- If Crypto API's are important to you over time, please keep an eye on the **Java Cryptographic Roadmap**
- This will help you **keep an eye on planned changes** in the Oracle JRE/JDK
- <http://www.java.com/cryptoroadmap/>
- Also consider 3<sup>rd</sup> party libraries that help developers make better cryptography API choices.
  - **Google KeyCzar** (Old School and Battle Hardened)
  - **Google TINK** (New and Shiny)
  - **LibSodium** (Best of Breed, Required Native APIs)

Honolulu



# Deprecation



## Web Plugin Deprecated in Java 9

- The Java Web Plugin will be deprecated in Java 9.
  - [https://blogs.oracle.com/java-platform-group/entry/moving\\_to\\_a\\_plugin\\_free](https://blogs.oracle.com/java-platform-group/entry/moving_to_a_plugin_free)
- It will still be there but will go away fully in a later version.
- Start moving to Java Web Start or javapackager.
- FUN FACT: Stuart Marks aka *Dr. Depreciator* is a member of the JDK team. See JEP 277
  - <https://twitter.com/DrDeprecator>
- `java.corba` is also deprecated in Java 9

Here's a few for the  
*Java Security Analysis Tool Geeks*

# Analyzing Java for Security

- A variety of JEP's In Java 9 will **help security tool vendors analyze Java** for security in more effective ways.
- **JEP-236**: Parser API for Nashorn (ECMAScript AST)
  - [https://blogs.oracle.com/java-platform-group/entry/nashorn\\_the\\_rhino\\_in\\_the](https://blogs.oracle.com/java-platform-group/entry/nashorn_the_rhino_in_the)
  - Delivered in Java 9
- **JEP-243**: Java-Level JVM Compiler Interface
  - <http://openjdk.java.net/jeps/243>
  - Delivered in Java 9
- **JEP-190**: Pluggable Static Analyzers
  - <http://openjdk.java.net/jeps/190>
  - Still in draft status

# JEP 200

## Java Modularity

# Why Reduce JRE Attack Surface?

- **Asset:** My application / runtime.
- **Threat:** Unknown future risk.
- **Mitigation:** Remove unused pieces.
- **How:** Compact Profiles (JDK 8) and Jigsaw (JDK 9)
- **Difficulty:** Removing things that you really do need will break your program.

## Server JRE Attack Surface Reduction

- **Server JRE decreases attack surface** by not including applets since **2013!**

# Savings from modularization in Java 8

- JDK 8 (embedded):
- Regular JRE: About 163MB.
  - Compact 3: Remove graphics, CORBA, and sound. About 21MB.
  - Compact 2: No Kerberos and JMX monitoring. About 15MB.
  - Compact 1: No JDBC and XML. About 11MB.



# JEP 200

## Java 9 Modularity (Jigsaw)

<http://openjdk.java.net/jeps/200>

- **Reliable configuration** to replace the classpath mechanism with a means for program components to declare dependences
- **Strong encapsulation** to allow a component to declare which of its public types are accessible to other components
- Decreases program size
- Reduces the attack surface by removing unused features
- <http://cr.openjdk.java.net/~mr/jigsaw/ea/module-summary.html> describes 72 **epic** levels of modularity.

JEP 290

# Filter Incoming Serialization Data

# Deserialization of Untrusted Data is Bad

- 2016 was the year of Java Deserialization apocalypse
  - Known vector since 2011 which allows RCE!
  - Previous lack of good RCE gadgets in common libraries
  - Apache Commons-Collections Gadget caught many off-guard
- Solution?
  - **Stop deserializing untrusted data**
  - Use a secure JSON/XML serializer instead

# JEP 290

## Filter Incoming Serialization Data

<http://openjdk.java.net/jeps/290>

- Allow **incoming streams of object-serialization data to be filtered** in order to improve security.
- Provide metrics to the **filter for graph size and complexity during deserialization** to validate normal graph behaviors.
- Provide a mechanism for **RMI-exported objects to validate the classes expected in invocations**.
- Define a **global filter that can be configured** by properties or a configuration file.

# ObjectInputFilter Interface

- Filter methods are **called during deserialization**
- **Validates classes** before deserialization
- **Validates array sizes**
- **Establish deserialization limits**
- Filter may **reject or be undecided**

## **NCC Group Whitepaper**

# **Combating Java Deserialization Vulnerabilities with Look-Ahead Object Input Streams (LAOIS)**

June 15, 2017

Prepared by

Robert C. Seacord

### **Abstract**

Java Serialization is an important and useful feature of Core Java that allows developers to transform a graph of Java objects into a stream of bytes for storage or transmission and then back into a graph of Java objects. Unfortunately, the Java Serialization architecture is highly insecure and has led to numerous vulnerabilities,

# Java 9 Filters

- Process-wide Filters
- Custom Filters
- Built-in Filters

*Java 9 adds additional methods to `ObjectInputStream` to set and get the current filter*

```
public class ObjectInputStream ... {  
  
    public final void  
    setObjectInputFilter(ObjectInputFilter filter);  
  
    public final ObjectInputFilter  
    getObjectInputFilter(ObjectInputFilter filter);  
  
}
```

*If no filter is set for an `ObjectInputStream` then the global filter is used, if any*



## Process Wide Filters

```
Properties props = System.getProperties();  
props.setProperty("jdk.serialFilter",  
"Bicycle;!*;maxdepth=1;maxrefs=1;maxbytes=78;  
maxarray=10");
```

- ***maxdepth*** maximum graph depth
- ***maxrefs*** maximum number of internal references
- ***maxbytes*** maximum input stream size
- ***maxarray*** maximum array length allowed

## *Process Wide Filters*

```
Properties props = System.getProperties();  
props.setProperty("jdk.serialFilter",  
"Bicycle;!*;maxdepth=1;maxrefs=1;maxbytes=78;  
maxarray=10");
```

*deserializes objects where the class name equals  
ser05j.Bicycle and **rejects all other classes***

# Custom Filters

```
class BikeFilter implements ObjectInputFilter {  
    private long maxStreamBytes = 78; // Maximum bytes in the stream.  
    private long maxDepth = 1; // Maximum depth of the graph allowed.  
    private long maxReferences = 1; // Maximum # of references in a graph.  
  
    @Override  
    public Status checkInput(FilterInfo filterInfo) {
```

Custom filters are created by implementing the `ObjectInputFilter` interface and overriding the `check-Input` method.

## *Built in Filters*

- RMI Registry and Distributed Garbage Collection (DGC) use JEP 290 Serialization Filtering to improve service security and robustness.
- RMI Registry and DGC implement built-in whitelist filters for the typical classes expected to be used with each service.

## **NCC Group Whitepaper**

# **Combating Java Deserialization Vulnerabilities with Look-Ahead Object Input Streams (LAOIS)**

June 15, 2017

Prepared by

Robert C. Seacord

### **Abstract**

Java Serialization is an important and useful feature of Core Java that allows developers to transform a graph of Java objects into a stream of bytes for storage or transmission and then back into a graph of Java objects. Unfortunately, the Java Serialization architecture is highly insecure and has led to numerous vulnerabilities,

# Deserialization and DOS

```
Properties props = System.getProperties();  
props.setProperty("jdk.serialFilter",  
"Bicycle;!*;maxdepth=1;maxrefs=1;maxbytes=78;  
maxarray=10");
```

- *Effectively prevents the SerialDOS exploit*
- *Vulnerable to generic heap DoS inside ObjectInputStream; heap DoS using nested Object[], ArrayList, and HashMap; collision attacks on Hashtable; and collision attacks on HashMap (Oracle Java 1.7)*

JEP 290 is available  
in Java 9, Java 8  
update 121, Java 7  
update 131, and Java  
6 update 141

# Remember?

## Deserialization of Untrusted Data is Bad

- 2016 was the year of Java Deserialization apocalypse
  - Known vector since 2011
  - Previous lack of good RCE
  - Apache Commons-Collection [Guard](#) [Guard](#)
- Solution?
  - Stop deserializing untrusted data
  - Use a secure JSON/XML serializer instead



# THE HORROR IS NOT OVER



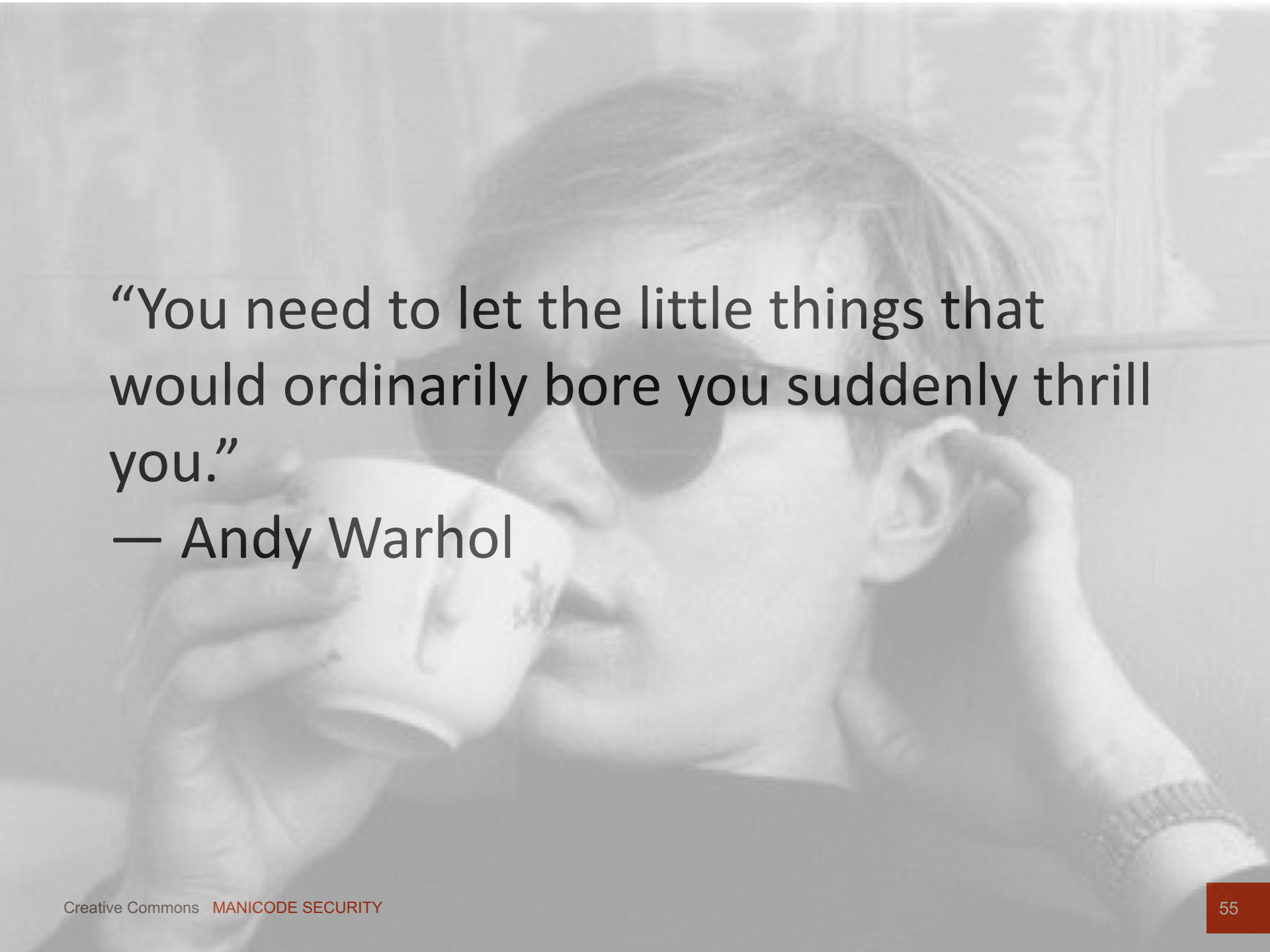
## Friday the 13<sup>th</sup>: JSON Attacks

Alvaro Muñoz (@pwntester)  
Oleksandr Mirosh

HPE Security



It's all about the little things



“You need to let the little things that would ordinarily bore you suddenly thrill you.”

— Andy Warhol

# The Little Things are Bigger Than They Appear



- Much of the **real work that has a big impact** is seeming boring stuff
- Take a look at the work from Joe Darcy
  - Cleaned up **thousands of bad lint warnings in the JDK**
  - Done by @SuppressWarnings or actually fixing the code
  - [https://blogs.oracle.com/darcy/entry/warnings\\_removal\\_advice](https://blogs.oracle.com/darcy/entry/warnings_removal_advice)
- A little **irrational exuberance** goes a long way

# Conclusion



# A BIG DUKE THANK YOU TO...

## Sean Mullan

- Technical lead and architect of Oracle's Java Security Libraries team.
- Lead of OpenJDK Security Group



# What can **you** do?



- **Create JEP's** for enhancements you would like to see in the JDK.
- **Support existing JEP's** you wish to see pushed live with comments, support and code!
- Check out the **Quality Outreach Campaign** which helps open source groups handle feedback
  - <https://wiki.openjdk.java.net/display/quality/Quality+Outreach>



*A hui ho, Java Ohana!*

`jim@manicode.com`