HTTPS certs making you certifiable?

Christopher M. Judd
ABOUT DOCKER

Almost overnight, Docker has become the de facto standard that developers and system administrators use for packaging, deploying, and running distributed applications. It provides tools for simplifying DevOps by enabling developers to create templates called images that can be used to create lightweight virtual machines called containers, which include their applications and all of their applications’ dependencies. These lightweight virtual machines can be promoted through testing and production environments where sysadmins deploy and run them. Docker makes it easier for organizations to automate infrastructure, isolate applications, maintain consistency, and improve resource utilizations.

Similar to the popular version control software Git, Docker has a social aspect, in that developers and sysadmins are able to share their images via Docker Hub.

Docker is an open-source solution that runs natively on Linux but also works on Windows and Mac using a lightweight Linux distribution and VirtualBox. Many tools have also grown up around Docker to make it easier to manage and orchestrate complex distributed applications.

DOCKER ARCHITECTURE

Docker utilizes a client-server architecture and a remote API to manage and create Docker containers built upon Linux containers. Docker containers are created from Docker images. The relationship between containers and images are analogous to the relationship between objects and classes in object-oriented programming.

Docker Images
A recipe or template for creating Docker containers. It includes the steps for installing and running the necessary software.

Docker Container
Like a tiny virtual machine that is created from the instructions found within the Docker image originated.

Docker Client
Command-line utility or other tool that takes advantage of the Docker API (https://docs.docker.com/reference/api/docker_remote_api) to communicate with a Docker daemon.

Docker Host
A physical or virtual machine that is running a Docker daemon and contains cached images as well as runnable containers created from images.

DOCKER MONITORING

Get Detailed Insight into Docker Containers

CONTENTS

Get More Refcardz!
Visit dzone.com/refcardz

© DZONE, INC.
DZONE.COM

BROUGHT TO YOU BY:

Christopher M. Judd
CTO and Partner at Manifest Solutions
Central Ohio Java Users Group leader
GET /en/ HTTP/1.1
Host: java.com
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.94 Safari/537.36
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8,pt;q=0.6
Cookie: s_cc=true; s_nr=1462828704344; gpName=javac%3AHompage; gpChannel=javac%3AHome; gpServer=java.com; s_sq=%5B%5B%5D%5D

HTTP/1.1 200 OK
Server: Oracle-Application-Server-11g
device_type: Any
host_service: FutureTenseContentServer:11.1.1.8.0
X-Powered-By: Servlet/2.5 JSP/2.1
Content-Type: text/html; charset=UTF-8
Content-Language: en
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Date: Mon, 09 May 2016 21:28:34 GMT
Content-Length: 2529
Connection: keep-alive

<html lang="en-US" xml:lang="en-US">
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <meta name="Language" content="en-US">
  <title>java.com: Java + You</title>
  <meta name="description" content="">
  <meta name="keywords" content="java, downloads, software">
  <meta name="date" content="2016-03-30">
</head>
</html>
GET /en/ HTTP/1.1
Host: java.com
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.94 Safari/537.36
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8,pt;q=0.6
Cookie: s_cc=true; s_nr=1462828704344; gpName=javac%3AHompage; gpChannel=javac%3AHome; gpServer=java.com; s_sq=%5B%5B%5D%5D

HTTP/1.1 200 OK
Server: Oracle-Application-Server-11g
device_type: Any
host_service: FutureTenseContentServer:11.1.1.8.0
X-Powered-By: Servlet/2.5 JSP/2.1
Content-Type: text/html; charset=UTF-8
Content-Language: en
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Date: Mon, 09 May 2016 21:28:34 GMT
Content-Length: 2529
Connection: keep-alive

<html lang="en-US" xml:lang="en-US"><head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<meta name="Language" content="en-US">
<title>java.com: Java + You</title>
<meta name="description" content="">
<meta name="keywords" content="java, downloads, software">
<meta name="date" content="2016-03-30">
</head>
<body>
</body></html>
[TCP Previous segment not captured] Encrypted Alert

Encrypted Application Data: 000000000003104ae4bb1e17aa3e58972a93d016dbcc9...
[TCP Previous segment not captured] Encrypted Alert

Java

https

Encrypted Application Data: 000000000003104ae4bb11e17aa3e58972a3d016dbcc9...
ALWAYS use HTTPS/SSL/TLS
Trust

Encryption
validate
The “Encryption” Debate

Posted on March 12, 2016

“Encryption” is quoted in the title of this essay because encryption is NOT what any of this is actually about. The debate is not about encryption, it’s about access. It should be called “The Device Access Debate” and encryption should have never been brought into it.

Modern smartphones have batteries, screens, memory, radios, encryption, and a bunch of other stuff. Collectively, they all make the phone go, they are all good, and we want as much of each as the device’s manufacturer can squeeze in. We do not want smaller batteries, lower resolution screens, less memory, less capable radios, or weaker encryption. And it is entirely proper for Apple to boast about the battery life, screen resolution, memory, radio, and encryption strength of their products. The FBI is entirely correct when it says that Apple is actively marketing the newly increased encryption strength of their latest phones and operating systems. That’s as is should be, in the same way that Apple is marketing their device’s battery life and screen resolution. Those are all features of modern smartphones, and Apple knows what their users want. We all want those things.

The fourth amendment to the US Constitution states: The right of the people to be secure in their persons, houses, papers, and effects, against unreasonable searches and seizures, shall not be violated, and no warrants shall issue, but upon probable cause, supported by
A2 – Broken Authentication and Session Management

Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users’ identities.
A6 – Sensitive Data Exposure
Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

A7 – Missing Function Level Access Control
Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.

A8 - Cross-Site Request Forgery (CSRF)
A CSRF attack forces a logged-on victim’s browser to send a forged HTTP request, including the victim’s session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim’s browser to generate requests the vulnerable application thinks are legitimate requests from the victim.

A9 - Using Components with Known Vulnerabilities
Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.

A10 – Unvalidated Redirects and Forwards
Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.
What’s Changed?
- new Certificate Authorities
- deprecated protocols and ciphers
- adoption of Java 8
SSL2
SSL3 \leq 128\text{ ciphers}
<table>
<thead>
<tr>
<th>TLS Protocols</th>
<th>JDK 8 (March 2014 to present)</th>
<th>JDK 7 (July 2011 to present)</th>
<th>JDK 6 (2006 to end of public updates 2013)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLSv1.2 (default)</td>
<td>TLSv1.2</td>
<td>TLS v1.1 (JDK 6 update 111 and above)</td>
<td></td>
</tr>
<tr>
<td>TLSv1.1</td>
<td>TLSv1.1</td>
<td>TLSv1 (default)</td>
<td></td>
</tr>
<tr>
<td>TLSv1</td>
<td>TLSv1 (default)</td>
<td>SSLv3</td>
<td></td>
</tr>
<tr>
<td>SSLv3</td>
<td>SSLv3</td>
<td>SSLv3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JSSE Ciphers:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ciphers in JDK 8</td>
<td>Ciphers in JDK 7</td>
<td>Ciphers in JDK 6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reference:</th>
<th>JDK 8 JSSE</th>
<th>JDK 7 JSSE</th>
<th>JDK 6 JSSE</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Java Cryptography Extension, Unlimited Strength (explained later)</th>
<th>JCE for JDK 8</th>
<th>JCE for JDK 7</th>
<th>JCE for JDK 6</th>
</tr>
</thead>
</table>


<table>
<thead>
<tr>
<th>TLS Protocols</th>
<th>JDK 8 (March 2014 to present)</th>
<th>JDK 7 (July 2011 to present)</th>
<th>JDK 6 (2006 to end of public updates 2013)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TLSv1.2 (default)</td>
<td>TLSv1.2</td>
<td>TLS v1.1 (JDK 6 update 111 and above)</td>
</tr>
<tr>
<td></td>
<td>TLSv1.1</td>
<td>TLSv1.1</td>
<td>TLSv1 (default)</td>
</tr>
<tr>
<td></td>
<td>TLSv1</td>
<td>TLSv1 (default)</td>
<td>TLSv1 (default)</td>
</tr>
<tr>
<td></td>
<td>SSLv3</td>
<td>SSLv3</td>
<td>SSLv3</td>
</tr>
<tr>
<td>JSSE Ciphers:</td>
<td>Ciphers in JDK 8</td>
<td>Ciphers in JDK 7</td>
<td>Ciphers in JDK 6</td>
</tr>
<tr>
<td>Reference:</td>
<td>JDK 8 JSSE</td>
<td>JDK 7 JSSE</td>
<td>JDK 6 JSSE</td>
</tr>
<tr>
<td>Java Cryptography</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extension, Unlimited</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strength (explained</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>later)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JCE for JDK 8</td>
<td>JCE for JDK 7</td>
<td>JCE for JDK 6</td>
</tr>
</tbody>
</table>
How do you debug HTTPS in Java?
curl ?
curl https://steve.grc.com/
curl https://steve.grc.com/
HOW WELL DO YOU KNOW SSL?

If you want to learn more about the technology that protects the Internet, you’ve come to the right place.

Books

*Bulletproof SSL and TLS* is a complete guide to deploying secure servers and web applications. This book, which provides comprehensive coverage of the ever-changing field of SSL/TLS and Web PKI, is intended for IT security professionals, system administrators, and developers, with the main focus on getting things done.  [MORE »](https://www.ssllabs.com/ssl-book/)

News

- **SSL Labs DROWN Test Implementation Details**
  March 4, 2016

- **DROWN Abuses SSL v2 to Attack TLS**
  March 1, 2016

- **Qualys WAF 2.0 New Feature**
  January 11, 2016

- **eDellRoot SSL Certificate Leaves Dell Endpoints At Risk to MITM Attacks**

About SSL Labs

SSL Labs is a collection of documents, tools and thoughts related to SSL. It’s an attempt to better understand how SSL is deployed, and an attempt to make it better. I hope that, in time, SSL Labs will grow into a forum where SSL will be discussed and improved.

SSL Labs is a non-commercial research effort, and we welcome participation from any individual and organization interested in SSL.

-- Ivan Ristić, Qualys
SSL Report: **steve.grc.com** (192.0.78.12)

**Summary**

**Overall Rating**

![A](image)

- **Certificate**: 100%
- **Protocol Support**: 100%
- **Key Exchange**: 100%
- **Cipher Strength**: 100%

Visit our [documentation page](#) for more information, configuration guides, and books. Known issues are documented [here](#).

This site works only in browsers with SNI support.

**Authentication**

**Server Key and Certificate #1**

<table>
<thead>
<tr>
<th>Subject</th>
<th>tls.automatic.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common names</td>
<td>tls.automatic.com</td>
</tr>
<tr>
<td>Alternative names</td>
<td>blog.grc.com, grc.nl, grcband.org, grcbizassurance.com, grcoe Xercisescienceclub.com, grc.org, grcdobunker.com</td>
</tr>
</tbody>
</table>
Chain issues: None

### Subject
- **Fingerprint SHA1:** e8a3b45b0d2d509e3582282d196e697d5996cb
- **Pin SHA256:** Ylh1t1fUR8y9Kjx30RhAn7kJkbGQgUvEwLbkBGF9Fu1hg=

**Valid until:** Wed, 17 Mar 2021 15:40:46 UTC (expires in 4 years and 10 months)

**Key:** RSA 2048 bits (e 65537)

**Issuer:** DST Root CA X3

**Signature algorithm:** SHA256withRSA

### Certification Paths

#### Path #1: Trusted
1. **Sent by server**
   - tis.automatic.com
     - **Fingerprint SHA1:** 903b5d4c75b338c0e25b601cd8b3c44f89d10c4
     - **Pin SHA256:** Bw71ZzSagqthaJfQcBiQWeiauJEEVmmU1SR+31com=
     - **RSA 2048 bits (e 65537) / SHA256withRSA**

2. **Sent by server**
   - Let's Encrypt Authority X3
     - **Fingerprint SHA1:** e8a3b45b0d2d509e3582282d196e697d5996cb
     - **Pin SHA256:** 7Yl7u1fUR8y9Kjx30RhAn7kJkbGQgUvEwLbkBGF9Fu1hg=
     - **RSA 2048 bits (e 65537) / SHA256withRSA**

3. **In trust store**
   - **DST Root CA X3** Self-signed
     - **Fingerprint SHA1:** dac9024f85ad8b549350c1732838ca8ac77c13
     - **Pin SHA256:** Vpshrz4i10wJncr1YKepWjgbo9rR63WwWxjMNN+eWysw=
     - **RSA 2048 bits (e 65537) / SHA1withRSA**

Weak or insecure signature, but no impact on root certificate

### Configuration

#### Protocols
- **TLS 1.2:** Yes
- **TLS 1.1:** Yes
- **TLS 1.0:** Yes
Configuration

### Protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS 1.2</td>
<td>Yes</td>
</tr>
<tr>
<td>TLS 1.1</td>
<td>Yes</td>
</tr>
<tr>
<td>TLS 1.0</td>
<td>Yes</td>
</tr>
<tr>
<td>SSL 3</td>
<td>No</td>
</tr>
<tr>
<td>SSL 2</td>
<td>No</td>
</tr>
</tbody>
</table>

### Cipher Suites (SSL 3+ suites in server-preferred order; deprecated and SSL 2 suites at the end)

<table>
<thead>
<tr>
<th>Cipher Suite</th>
<th>Key Exchange</th>
<th>Key Encryption</th>
<th>MAC</th>
<th>Initial</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0x001f)</td>
<td>ECDH secp256r1 (eq. 3072 bits RSA)</td>
<td>AES 128 GCM</td>
<td>SHA256</td>
<td>FS</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0x0030)</td>
<td>ECDH secp384r1 (eq. 3072 bits RSA)</td>
<td>AES 256 GCM</td>
<td>SHA384</td>
<td>FS</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0x0027)</td>
<td>ECDH secp256r1 (eq. 3072 bits RSA)</td>
<td>AES 128 CBC</td>
<td>SHA256</td>
<td>FS</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0x0013)</td>
<td>ECDH secp256r1 (eq. 3072 bits RSA)</td>
<td>AES 128 CBC</td>
<td>SHA</td>
<td>FS</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0x0028)</td>
<td>ECDH secp384r1 (eq. 3072 bits RSA)</td>
<td>AES 256 CBC</td>
<td>SHA384</td>
<td>FS</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0x0014)</td>
<td>ECDH secp256r1 (eq. 3072 bits RSA)</td>
<td>AES 256 CBC</td>
<td>SHA</td>
<td>FS</td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)</td>
<td>RSA</td>
<td>AES 128 GCM</td>
<td>SHA256</td>
<td>FS</td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)</td>
<td>RSA</td>
<td>AES 256 GCM</td>
<td>SHA384</td>
<td>FS</td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_128_CBC_SHA256 (0x003c)</td>
<td>RSA</td>
<td>AES 128 CBC</td>
<td>SHA256</td>
<td>FS</td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)</td>
<td>RSA</td>
<td>AES 128 CBC</td>
<td>SHA</td>
<td>FS</td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_256_CBC_SHA384 (0x003d)</td>
<td>RSA</td>
<td>AES 256 CBC</td>
<td>SHA384</td>
<td>FS</td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)</td>
<td>RSA</td>
<td>AES 256 CBC</td>
<td>SHA</td>
<td>FS</td>
</tr>
</tbody>
</table>
### Protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS 1.2</td>
<td>Yes</td>
</tr>
<tr>
<td>TLS 1.1</td>
<td>Yes</td>
</tr>
<tr>
<td>TLS 1.0</td>
<td>Yes</td>
</tr>
<tr>
<td>SSL 3</td>
<td>No</td>
</tr>
<tr>
<td>SSL 2</td>
<td>No</td>
</tr>
</tbody>
</table>

### Cipher Suites (SSL 3+ suites in server-preferred order; deprecated and SSL 2 suites at the end)

<table>
<thead>
<tr>
<th>Cipher Suite</th>
<th>Message Authentication</th>
<th>Key Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)</td>
<td>ECDH secp256r1 (req. 3072 bits RSA)</td>
<td>FS</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)</td>
<td>ECDH secp256r1 (req. 3072 bits RSA)</td>
<td>FS</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)</td>
<td>ECDH secp256r1 (req. 3072 bits RSA)</td>
<td>FS</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)</td>
<td>ECDH secp256r1 (req. 3072 bits RSA)</td>
<td>FS</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)</td>
<td>ECDH secp256r1 (req. 3072 bits RSA)</td>
<td>FS</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)</td>
<td>ECDH secp256r1 (req. 3072 bits RSA)</td>
<td>FS</td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_256_GCM_SHA384 (0x9d)</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_128_CBC_SHA256 (0x93)</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_256_CBC_SHA (0x94)</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_WITH_3DES_EDE_CBC_SHA (0xa)</td>
<td>112</td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_WITH_CAMELLIA_256_CBC_SHA (0x84)</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_WITH_CAMELLIA_128_CBC_SHA (0x81)</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (0xc012)</td>
<td>ECDH secp256r1 (req. 3072 bits RSA)</td>
<td>FS</td>
</tr>
</tbody>
</table>

### Handshake Simulation

<table>
<thead>
<tr>
<th>Version</th>
<th>Support</th>
<th>Incorrect Certificate Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android 2.3.7</td>
<td>No SNI²</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>Android 4.0.4</td>
<td>RSA 2048 (SHA256)</td>
<td>TLS 1.0</td>
</tr>
<tr>
<td>Android 4.1.1</td>
<td>RSA 2048 (SHA256)</td>
<td>TLS 1.0</td>
</tr>
<tr>
<td>Android 4.2.2</td>
<td>RSA 2048 (SHA256)</td>
<td>TLS 1.0</td>
</tr>
<tr>
<td>Device</td>
<td>Protocol</td>
<td>Version</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>Android 2.3.7</td>
<td>TLS 1.0</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>Android 4.0.4</td>
<td>TLS 1.0</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>Android 4.1.1</td>
<td>TLS 1.0</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>Android 4.2.2</td>
<td>TLS 1.0</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>Android 4.3</td>
<td>TLS 1.0</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>Android 4.4.2</td>
<td>TLS 1.2</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>Android 5.0.0</td>
<td>TLS 1.2</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>Baidu Jan 2015</td>
<td>TLS 1.0</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>BisPreview Jan 2015</td>
<td>TLS 1.2</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>Chrome 48 / OS X</td>
<td>TLS 1.2</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>Firefox 31.3.0 ESR / Win 7</td>
<td>TLS 1.2</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>Firefox 42 / OS X</td>
<td>TLS 1.2</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>Firefox 44 / OS X</td>
<td>TLS 1.2</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>Googletbot Feb 2015</td>
<td>TLS 1.2</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>IE 6 / XP</td>
<td>TLS 1.0</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>IE 7 / Vista</td>
<td>TLS 1.0</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>IE 8 / XP</td>
<td>TLS 1.0</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>IE 8-10 / Win 7</td>
<td>TLS 1.0</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>IE 11 / Win 7</td>
<td>TLS 1.2</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>IE 11 / Win 8.1</td>
<td>TLS 1.2 &gt; http/1.1</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>IE 10 / Win Phone 8.0</td>
<td>TLS 1.0</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>IE 11 / Win Phone 8.1</td>
<td>TLS 1.2 &gt; http/1.1</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>IE 11 / Win Phone 8.1 Update</td>
<td>TLS 1.2 &gt; http/1.1</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>IE 11 / Win 10</td>
<td>TLS 1.2 &gt; http/1.1</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>Edge 13 / Win 10</td>
<td>TLS 1.2 &gt; http/1.1</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>Edge 13 / Win Phone 10</td>
<td>TLS 1.2 &gt; http/1.1</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>Java 6u45</td>
<td>TLS 1.0</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>Java 7u25</td>
<td>TLS 1.0</td>
<td>RSA 2048 (SHA256)</td>
</tr>
<tr>
<td>Protocol Details</td>
<td>Status</td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------------</td>
<td></td>
</tr>
<tr>
<td><strong>DROWN (experimental)</strong></td>
<td>No, server keys and hostname not seen elsewhere with SSLv2</td>
<td></td>
</tr>
<tr>
<td>(1) For a better understanding of this test, please read <a href="#">this longer explanation</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2) Key usage data kindly provided by the Censys network search engine; original DROWN test <a href="#">here</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3) Censys data is only indicative of possible key and certificate reuse; possibly out-of-date and not complete</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Secure Renegotiation</strong></td>
<td>Supported</td>
<td></td>
</tr>
<tr>
<td><strong>Secure Client-Initiated Renegotiation</strong></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><strong>Insecure Client-Initiated Renegotiation</strong></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><strong>BEAST attack</strong></td>
<td>Not mitigated server-side (<a href="#">more info</a>)</td>
<td></td>
</tr>
<tr>
<td><strong>POODLE (SSLv3)</strong></td>
<td>No, SSL 3 not supported (<a href="#">more info</a>)</td>
<td></td>
</tr>
<tr>
<td><strong>POODLE (TLS)</strong></td>
<td>No (<a href="#">more info</a>)</td>
<td></td>
</tr>
<tr>
<td><strong>Downgrade attack prevention</strong></td>
<td>Yes, TLS_FALLBACK_SCSV supported (<a href="#">more info</a>)</td>
<td></td>
</tr>
<tr>
<td><strong>SSL/TLS compression</strong></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><strong>RC4</strong></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><strong>Heartbeat (extension)</strong></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td><strong>Heartbleed (vulnerability)</strong></td>
<td>No (<a href="#">more info</a>)</td>
<td></td>
</tr>
<tr>
<td><strong>OpenSSL CCS vuln. (CVE-2014-0224)</strong></td>
<td>No (<a href="#">more info</a>)</td>
<td></td>
</tr>
<tr>
<td><strong>Forward Secrecy</strong></td>
<td>With modern browsers (<a href="#">more info</a>)</td>
<td></td>
</tr>
<tr>
<td><strong>ALPN</strong></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td><strong>NPN</strong></td>
<td>Yes h2 http/1.1</td>
<td></td>
</tr>
<tr>
<td><strong>Session resumption (caching)</strong></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td><strong>Session resumption (tickets)</strong></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td><strong>OCSP stapling</strong></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><strong>Strict Transport Security (HSTS)</strong></td>
<td>Yes, TOO SHORT (less than 180 days) <a href="#">max-age=86400</a></td>
<td></td>
</tr>
<tr>
<td><strong>HSTS Preloading</strong></td>
<td>Not in: Chrome Edge Firefox IE Tor</td>
<td></td>
</tr>
<tr>
<td><strong>Public Key Pinning (HPKP)</strong></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><strong>Public Key Pinning Report-Only</strong></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><strong>Long handshake intolerance</strong></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><strong>TLS extension intolerance</strong></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><strong>TLS version intolerance</strong></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><strong>Incorrect SNI alerts</strong></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><strong>Uses common DH primes</strong></td>
<td>No, DHE suites not supported</td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------</td>
<td></td>
</tr>
<tr>
<td>ALPN</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NPN</td>
<td>Yes, h2 http/1.1</td>
<td></td>
</tr>
<tr>
<td>Session resumption (caching)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Session resumption (tickets)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>OCSP stapling</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><strong>Strict Transport Security (HSTS)</strong></td>
<td><strong>Too short (less than 180 days)</strong></td>
<td></td>
</tr>
<tr>
<td>HSTS Preloading</td>
<td>Not in: Chrome Edge Firefox IE Tor</td>
<td></td>
</tr>
<tr>
<td>Public Key Pinning (HPKP)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Public Key Pinning Report-Only</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Long handshake intolerance</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>TLS extension intolerance</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>TLS version Intolerance</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Incorrect SNI alerts</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Uses common DH primes</td>
<td>No, DHE suites not supported</td>
<td></td>
</tr>
<tr>
<td>DH public server param (Yes) reuse</td>
<td>No, DHE suites not supported</td>
<td></td>
</tr>
<tr>
<td>SSL 2 handshake compatibility</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

**Miscellaneous**

- **Test date**: Tue, 10 May 2016 01:56:42 UTC
- **Test duration**: 70,249 seconds
- **HTTP status code**: 200
- **HTTP server signature**: nginx
- **Server hostname**: -
Java implementation of a "cURL like" (Client URL Request Library) software

Jean-Yves CRONIER Windows bat to launch jcurl
src Handle session cookie a month ago
.gitignore first commit 3 years ago
LICENSE first commit 3 years ago
README.md prise en charge des réponses HTTP GZIP et correction des caractères a... 3 years ago
TODO.md Create TODO.md 2 years ago
jcurl.bat Windows bat to launch jcurl a month ago
pom.xml prise en charge des réponses HTTP GZIP et correction des caractères a... 3 years ago

README.md

jcURL

Java Client URL Request Library - Java implementation of a cURL (Client URL Request Library) like tool.

Synopsis

java -jar jcurl.jar [options] [URL...]

https://github.com/jycr/jcurl
java -jar jcurl.jar https://steve.grc.com/
Caused by: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target
  at sun.security.provider.certpath.SunCertPathBuilder.engineValidate(PKIXValidator.java:292)
  at sun.security.provider.certpath.SunCertPathBuilder.engineValidate(PKIXValidator.java:260)
  at sun.security.x509.TrustManagerImpl.validate(X509TrustManagerImpl.java:324)
  at sun.security.x509.TrustManagerImpl.checkTrusted(X509TrustManagerImpl.java:229)
  at sun.security.x509.TrustManagerImpl.checkServerTrusted(X509TrustManagerImpl.java:124)
  at sun.security.x509.TrustManagerImpl.doBuild(X509TrustManagerImpl.java:145)
  ... 21 more

Caused by: java.net.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target
  at sun.security.provider.certpath.SunCertPathBuilder.engineValidate(PKIXValidator.java:387)
  at sun.security.provider.certpath.SunCertPathBuilder.engineValidate(PKIXValidator.java:260)
  at sun.security.x509.TrustManagerImpl.validate(X509TrustManagerImpl.java:324)
  at sun.security.x509.TrustManagerImpl.checkTrusted(X509TrustManagerImpl.java:229)
  at sun.security.x509.TrustManagerImpl.checkServerTrusted(X509TrustManagerImpl.java:124)
  at sun.security.x509.TrustManagerImpl.doBuild(X509TrustManagerImpl.java:145)
  ... 27 more
-Djavax.net.debug=ssl:handshake:verbose
java -Djavax.net.debug=ssl:handshake:verbose -jar jcurl.jar https://steve.grc.com/
keyStore is:
keyStore type is: jks
keyStore provider is:
init Keystore

trustStore is: /usr/java/jdk1.8.0_25/jre/lib/security/cacerts
trustStore type is: jks
trustStore provider is:
init truststore

adding as trusted cert:
Subject: CN=SecureTrust CA, O=SecureTrust Corporation, C=US
Issuer: CN=SecureTrust CA, O=SecureTrust Corporation, C=US
Algorithm: RSA; Serial number: 0xcf08e5e5e6a5ad5427f10eb271059d0
Valid from Tue Nov 07 19:31:18 UTC 2006 until Mon Dec 31 19:40:55 UTC 2029

adding as trusted cert:
Subject: CN=Starfield Root Certificate Authority - G2, O="Starfield Technologies, Inc.", L=Scottsdale, ST=Arizona, C=US
Issuer: CN=Starfield Root Certificate Authority - G2, O="Starfield Technologies, Inc.", L=Scottsdale, ST=Arizona, C=US
Algorithm: RSA; Serial number: 0x0
Valid from Tue Sep 01 00:00:00 UTC 2009 until Thu Dec 31 23:59:59 UTC 2037

adding as trusted cert:
Subject: CN=VeriSign Class 2 Public Primary Certification Authority - G3, OU="(c) 1999 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US
Issuer: CN=VeriSign Class 2 Public Primary Certification Authority - G3, OU="(c) 1999 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US
Algorithm: RSA; Serial number: 0x6170cb498c5f94529e7ba6d950b7a
Valid from Fri Oct 01 00:00:00 UTC 1999 until Wed Jul 16 23:59:59 UTC 2036

adding as trusted cert:
Subject: CN=Security Communication RootCA1, O=SECOM Trust.net, C=JP
Issuer: CN=Security Communication RootCA1, O=SECOM Trust.net, C=JP
Algorithm: RSA; Serial number: 0x0
Valid from Tue Sep 30 04:20:49 UTC 2003 until Sat Sep 30 04:20:49 UTC 2023

adding as trusted cert:
Subject: CN=DigiCert Global Root CA, OU=www.digicert.com, O=DigiCert Inc, C=US
Issuer: CN=DigiCert Global Root CA, OU=www.digicert.com, O=DigiCert Inc, C=US
Algorithm: RSA; Serial number: 0x83be056904246b1a1756ac95991c74a
Valid from Fri Nov 10 00:00:00 UTC 2006 until Mon Nov 10 00:00:00 UTC 2031

adding as trusted cert:
Subject: CN=Entrust.net Certification Authority (2048), OU="(c) 1999 Entrust.net Limited, O=www.entrust.net/CPS_2048 incorp. by ref. (limits liab.), 0=Entrust.net
Issuer: CN=Entrust.net Certification Authority (2048), OU="(c) 1999 Entrust.net Limited, O=www.entrust.net/CPS_2048 incorp. by ref. (limits liab.), 0=Entrust.net
Algorithm: RSA; Serial number: 0x3633def8
Valid from Fri Dec 24 17:50:51 UTC 1999 until Tue Jul 24 14:15:12 UTC 2029

adding as trusted cert:
Subject: CN=Equifax Secure eBusiness CA-1, O=Equifax Secure Inc., C=US
Issuer: CN=Equifax Secure eBusiness CA-1, O=Equifax Secure Inc., C=US
Algorithm: RSA; Serial number: 0x4
Valid from Mon Jun 21 04:00:00 UTC 1999 until Sun Jun 21 04:00:00 UTC 2020

adding as trusted cert:
Subject: CN=thawte Primary Root CA, OU="(c) 2006 thawte, Inc. - For authorized use only", OU=Certification Services Division, O="thawte, Inc.", C=US
Issuer: CN=thawte Primary Root CA, OU="(c) 2006 thawte, Inc. - For authorized use only", OU=Certification Services Division, O="thawte, Inc.", C=US
Algorithm: RSA; Serial number: 0x344ed557203ed6e49f42f36eb7d2b6d
Valid from Fri Nov 17 00:00:00 UTC 2006 until Wed Jul 16 23:59:59 UTC 2036

adding as trusted cert:
Subject: EMAILADDRESS=info@valicert.com, CN=http://www.valicert.com/, O=ValiCert Class 2 Policy Validation Authority, O=ValiCert, Inc.", L=ValiCert Validation Network
Issuer: EMAILADDRESS=info@valicert.com, CN=http://www.valicert.com/, O=ValiCert Class 2 Policy Validation Authority, O=ValiCert, Inc.", L=ValiCert Validation Network
Algorithm: RSA; Serial number: 0x1
Valid from Sat Jun 26 00:19:54 UTC 1999 until Wed Jun 26 00:19:54 UTC 2019

adding as trusted cert:
Subject: CN=Go Daddy Root Certificate Authority - G2, O=GoDaddy.com, Inc., L=Scottsdale, ST=Arizona, C=US
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target

Caused by: javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target

Caused by: javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target
[4]: ObjectId: 2.5.29.31 Criticality=false
CRLDistributionPoints [
  [DistributionPoint:
    [URIName: http://crl.identrust.com/DSTROOTCA3CRL.crl]]
]

[5]: ObjectId: 2.5.29.32 Criticality=false
CertificatePolicies [
  [CertificatePolicyId: [2.23.140.1.2]]
  [CertificatePolicyId: [1.3.6.1.4.1.44947.1.1.1]]
  [PolicyQualifierInfo:
    qualifierID: 1.3.6.1.5.5.7.2.1
    qualifier: 0000: 16 22 68 74 70 3A 2F 2F 63 70 73 2E 72 6F 6F  ."http://cps.roo
0010: 74 2D 78 31 2E 6C 65 74 73 65 6E 63 72 79 70 74  .t-x1.letsencrypt
0020: 2E 6F 72 67 .org]
]

[6]: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
  DigitalSignature
  KeyCertSign
  Crl_Sign
]

[7]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
  KeyIdentifier [
    0000: A8 4A 6A 63 04 7D DD BA .Jjc......9..Ee.
    0010: F3 A8 EC A1 ....
  ]
  Algorithm: [SHA256withRSA]
  Signature:
    0000: DD 33 D7 11 F3 63 58 38 .3...cX8...U.v
    0010: 56 B9 70 48 A5 69 47 27 .v.ph.IG'...Z.
    0020: 4A 12 37 24 74 51 CD 9B .A...q...Z.
    0030: F7 A4 BC 4E 20 5C 4B 98 .eb...\K...
    0040: 87 B9 06 2D 2C 68 26 C6 .X../,h&.K....
    0050: 0B ED 43 12 12 44 6E 73 7A ...C.Dnosz...n.
    0060: 5C 7D 87 DD E0 C9 02 44 ...\......D....D
    0070: A7 87 AF 7D 87 DD E0 C9 02 44 ...L......D....D
    0080: 87 BD 06 2F 2C 68 26 C6 .X../,h&.K....
    0090: 0B ED 43 12 12 44 6E 73 7A ...C.Dnosz...n.
    00A0: 5C 7D 87 DD E0 C9 02 44 ...\......D....D
    00B0: 87 BD 06 2F 2C 68 26 C6 .X../,h&.K....
    00C0: 0B ED 43 12 12 44 6E 73 7A ...C.Dnosz...n.
    00D0: 5C 7D 87 DD E0 C9 02 44 ...\......D....D
]

*** Invalidated: [Session-1, TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256] main, SEND TLSv1.2 ALERT: fatal, description = certificate_unknown
main, WRITE: TLSv1.2 Alert, length = 2
main, called closeSocket()
# Certification Paths

**Path #1: Trusted**

1. **Sent by server**
   - **tls.automattic.com**
     - Fingerprint SHA1: f903b5f4b79b83c0a2a256801c8e3c44f5fd10d40
     - Pin SHA256: Bw7I2Zn8agrdi6aJFqCB0wVllaeurJE6VmmJ1SR+/31qc=
     - RSA 2048 bits (e 65537) / SHA256withRSA

2. **Sent by server**
   - **Let's Encrypt Authority X3**
     - Fingerprint SHA1: e6a3b45b062d509b3382282d19e6e975958cc5
     - Pin SHA256: YLh1dUR9y6Kja30RrAn7JKmQgQqE8LMk6gF2Fuihg=
     - RSA 2048 bits (e 65537) / SHA256withRSA

3. **In trust store**
   - **DST Root CA X3** **Self-signed**
     - Fingerprint SHA1: dac9024f54d86f94935fb1732638ca6ad77c13
     - Pin SHA256: Vja8r4z+80xjNcr1YKepWQboSIR63rWxVlMN+eWy8=
     - RSA 2048 bits (e 65537) / SHA1withRSA
      
           Weak or insecure signature, but no impact on root certificate
**Certificate Chain Download Instructions**

Before you can use your digital certificate, you must install the Root Certificate Chain on your server. This consists of two steps:

1. Download the required Root Certificate
2. Install the Root Certificate

Print these instructions. Then select each of the links below to download all of the required root certificates. Follow the instructions to navigate through the process of installing the Root Certificates.

DST Root CA X3
DST Root CA A5

1. For each of the links presented above, when you select the link, the File Download window displays. Complete the following action:

   **Action**
   
a) Copy and paste the contents of the certificate, including the "-----BEGIN CERTIFICATE-----" and "-----END CERTIFICATE-----" into a text file.

   b) Save the file as *certificate.pem* on your computer. (*Note: Replace certificate with the name of the link above.*)

   c) Complete this step for all DST Root links above.

   Your Download is complete!

2. Install the Root Certificates by completing the following actions for each of the files saved above:

   **Action**
   
a) Connect to the NES admin page (http://...).

   b) Click Keys and Certificate.

   c) Click Install Certificate.

   d) In the page that appears, select Trusted Certificate Authority (CA).

   e) In the Certificate Name field, enter the name of the certificate, e.g. DST Root CA.

   f) Select the Message is in this file radio button, then enter the path and the certificate.pem file you downloaded previously.

   g) Select the alias for the key pair you used to make the request, then click Ok. The Add Server Certificate screen appears.

   h) Verify the information on the certificate. If everything looks good, click the Add Certificate button.

   i) A warning message appears that states, “Security changes require shutdown.” Click Ok to continue.

   j) A message appears that states, “Success! Your server certificate has been installed. You may now turn on encryption for your server!” Click Ok to continue.

   k) Complete these instructions for all *certificate.pem* files saved above.

   Congratulations! Your download is complete.

Note: You may also refer to your Web Server Installation instructions for details.
ROOT CHAIN INSTALL

Copy and paste the following DST Root certificate into a text file on your computer. Select the back button on your browser to return to the previous instructions page and continue with your install process.

```
MIIDCCAKgAvIiBAgIBAgIQERjqJMDmGLVhA9aZuANBqghk/G9wOBABQUFADAJ/
MSqWlYDQVQKExEaaWdpGFlsIFNpZ55hVLyZ58BwCvZ8DDby4xZ2FZwVgNV
BAMT
DkRTVC8Bsb29oiENF1Fq9iM84XDTAwmDkzMDmzMTixOvO7DTe1XkzmMD78MDEx
NVow
PzExMiCA1UEhMoRGAhNCBAEBMBhhcBTAWIuXyRfcmLgyVJk13Qg283MbMwFQQ
DVQjD
Ew5EUIQgmW9vCmDBQDGDMzByMzCASA1wQyJKzKizwNAGEBBQAdgEPADCCAQ
ocQgEB
AN+v6ZcQINE1MxIZf9qqu2ZbxyRm7Pz83Dc2AwRjUI+DoM3ZJkmuILUm7R
4d
rzSly2Xu/NMhD2Xskky4zJ93ewEtu1Cj68b87XMuegGMO1if0oUMM80ReOq
OL15Cj9HUL2Azd+3UW0DyOK1yEPyLYHhUmouOLJiGl5KQoDmNjOjoJg4XLh87dn9
b
x/kOky69cK3FCx0/kHyxTzqoz2WIMn/5WgTe1OLyNau7FCokh49Z1QXm+y/JUfW
7BZy1bsOFUQ9D8/RnhQPGx89Wam400utocub3YE8VJAr2m7Xp/71XAcPnAD
aeQomxkt/lX4+U9m5/wJ0CawEAaANCMEAwDwYDVR0TAQH/BAUw AwEw/zAO
BgNV
HQBBAfE8EBBAMCAQwGHQVDV0R08BBYEFMSnRs7RHLH82+FLKHX/BVgHyYkGMA
0GCsQgG
Stb3DQEBAQUA4iBAQ0Cj6yFwBqcR7uKGY30r+Dxz9LwWmmbSBd49zZRNi+D7
69
ikg9D8/0EJcdBd0f9q3A3T5MgRSP6cz6F93bXanx5+5Vg3t23A9D1cmEmv BuXr
AvhRaoszYtQQXejB5G7YGBeArwDGpXRCnWYALbRm9Y9Kl+rlmM6pZB7W8px7
Zz
R8srzJmwnOj0/P4ILz9c8PDHyh8bwRLtGm1D9jSZeMmtm7r/md20xZjbA3JWFBM5
J5DGfogCWjBJ4d1q37wCCZAABA2RjTjSv9jjeEgSzZ7LT0DyjWW68Xyy93beqbo
Y0
Ob8VZzj9neWagqNdwwVYkQsEjgfKbYK7p2C7NTUQ
```
$JAVA_HOME/bin/keytool -trustcacerts -keystore $JAVA_HOME/jre/lib/security/cacerts -storepass changeit -noprompt -importcert -file dstrootcax3.pem
java -Djavax.net.debug=ssl:handshake:verbose -jar jcurl.jar https://steve.grc.com/
```
-Dhttps.protocols=TLSv1,TLSv1.1,TLSv1.2
```
SSL Report: [Redacted]

Assessed on: Tue, 10 May 2016 01:19:08 UTC | Hide | Clear cache

Summary

Overall Rating

Certificate
Protocol Support
Key Exchange
Cipher Strength

Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This server supports SSL 2, which is obsolete and insecure, and can be used against TLS (DROWN attack). Grade set to F. MORE INFO »

This server supports 512-bit export suites and might be vulnerable to the FREAK attack. Grade set to F. MORE INFO »

This server is vulnerable to the POODLE TLS attack. Patching required. Grade set to F. MORE INFO »

This server uses SSL 3, which is obsolete and insecure. Grade capped to B. MORE INFO »

The server supports only older protocols, but not the current best TLS 1.2. Grade capped to C. MORE INFO »

This server accepts RC4 cipher, but only with older protocol versions. Grade capped to B. MORE INFO »

There is no support for secure renegotiation. MORE INFO »
Configuration

Protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS 1.2</td>
<td>No</td>
</tr>
<tr>
<td>TLS 1.1</td>
<td>No</td>
</tr>
<tr>
<td>TLS 1.0</td>
<td>Yes</td>
</tr>
<tr>
<td>SSL 3</td>
<td>INSECURE</td>
</tr>
<tr>
<td>SSL 2</td>
<td>INSECURE</td>
</tr>
</tbody>
</table>

Cipher Suites (SSL 3+ suites in server-preferred order; deprecated and SSL 2 suites at the end)

<table>
<thead>
<tr>
<th>Cipher Suite</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_RSA_WITH_RC4_128_MD5 (0x4)</td>
<td>INSECURE</td>
</tr>
<tr>
<td>TLS_RSA_WITH_RC4_128_SHA (0x3)</td>
<td>INSECURE</td>
</tr>
<tr>
<td>TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x3)</td>
<td>WEAK</td>
</tr>
<tr>
<td>TLS_RSA_WITH_DES_CBC_SHA (0x9)</td>
<td>WEAK</td>
</tr>
<tr>
<td>TLS_RSA_EXPORT1024_WITH_RC4_56_SHA (0x64)</td>
<td>INSECURE</td>
</tr>
<tr>
<td>TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA (0x62)</td>
<td>WEAK</td>
</tr>
<tr>
<td>TLS_RSA_EXPORT_WITH_RC4_40_MD5 (0x3)</td>
<td>INSECURE</td>
</tr>
<tr>
<td>TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 (0x8)</td>
<td>INSECURE</td>
</tr>
<tr>
<td>SSL_CK_RC2_128_CBC_EXPORT40_WITH_MD5 (0x00080)</td>
<td>INSECURE</td>
</tr>
<tr>
<td>SSL_CK_RC4_128_EXPORT40_WITH_MD5 (0x00080)</td>
<td>INSECURE</td>
</tr>
<tr>
<td>SSL_CK_DES_64_CBC_WITH_MD5 (0x60040)</td>
<td>INSECURE</td>
</tr>
<tr>
<td>SSL_CK_RC2_128_CBC_WITH_MD5 (0x80080)</td>
<td>INSECURE</td>
</tr>
<tr>
<td>SSL_CK_DES_192_EDE_CBC_WITH_MD5 (0x70000)</td>
<td>INSECURE</td>
</tr>
<tr>
<td>SSL_CK_RC4_128_WITH_MD5 (0x10080)</td>
<td>INSECURE</td>
</tr>
</tbody>
</table>

Handshake Simulation

<table>
<thead>
<tr>
<th>Platform</th>
<th>Feature</th>
<th>Version</th>
<th>SSL 3.0</th>
<th>TLS 1.0</th>
<th>TLS_RSA_WITH_RC4_128_MD5</th>
<th>Fallback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android 2.3.7</td>
<td>No SNI 2</td>
<td>RSA 2048 (SHA256)</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>RC4</td>
</tr>
<tr>
<td>Java</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protocol Details</td>
<td>Status</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>--------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>DROWN (experimental)</strong></td>
<td>No, server keys and hostname not seen elsewhere with SSLv2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(1) For a better understanding of this test, please read <a href="#">this longer explanation</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(2) Key usage data kindly provided by the Censys network search engine; original DROWN test <a href="#">here</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(3) Censys data is only indicative of possible key and certificate reuse; possibly out-of-date and not complete</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Secure Renegotiation</strong></td>
<td>Not supported</td>
<td>ACTION NEEDED (more info)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Secure Client-Initiated Renegotiation</strong></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Insecure Client-Initiated Renegotiation</strong></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BEAST attack</strong></td>
<td>Not mitigated server-side (more info)</td>
<td>SSL 3: 0x6, TLS 1.0: 0x6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>POODLE (SSLv3)</strong></td>
<td>No, mitigated (more info)</td>
<td>SSL 3: 0x6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>POODLE (TLS)</strong></td>
<td>Vulnerable</td>
<td>INSECURE (more info)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Downgrade attack prevention</strong></td>
<td>No, TLS_FALLBACK_SCSV not supported (more info)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SSL/TLS compression</strong></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RC4</strong></td>
<td>Yes</td>
<td>INSECURE (more info)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Heartbeat (extension)</strong></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Heartbleed (vulnerability)</strong></td>
<td>No (more info)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OpenSSL CCS vuln. (CVE-2014-0224)</strong></td>
<td>No (more info)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Forward Secrecy</strong></td>
<td>No</td>
<td>WEAK (more info)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALPN</strong></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NPN</strong></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Session resumption (caching)</strong></td>
<td>No (IDs assigned but not accepted)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Session resumption (tickets)</strong></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OCSP stapling</strong></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Strict Transport Security (HSTS)</strong></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>HSTS Preloading</strong></td>
<td>Not In: Chrome Edge Firefox IE Tor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Public Key Pinning (HPKP)</strong></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Public Key Pinning Report-Only</strong></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Long handshake intolerance</strong></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TLS extension intolerance</strong></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TLS version intolerance</strong></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Incorrect SNI alerts</strong></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Uses common DH primes</strong></td>
<td>No, DHE suites not supported</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>DH public server param (Xe) reuse</strong></td>
<td>No, DHE suites not supported</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Self Signing
$JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA

Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]: Chris Judd
What is the name of your organizational unit?
  [Unknown]: Juddi
What is the name of your organization?
  [Unknown]: Manifest Solutions
What is the name of your City or Locality?
  [Unknown]: Columbus
What is the name of your State or Province?
  [Unknown]: OH
What is the two-letter country code for this unit?
  [Unknown]: US
Is CN=Chris Judd, OU=Juddi, O=Manifest Solutions, L=Columbus, ST=OH, C=US correct?
  [no]: yes

Enter key password for <tomcat>
  (RETURN if same as keystore password):
Re-enter new password:

~/.keystore
keytool -list -v

Enter keystore password:

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: tomcat
Creation date: May 10, 2016
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Chris Judd, OU=Juddi, O=Manifest Solutions, L=Columbus, ST=OH, C=US
Issuer: CN=Chris Judd, OU=Juddi, O=Manifest Solutions, L=Columbus, ST=OH, C=US
Serial number: 42e9ae25
Certificate fingerprints:
Signature algorithm name: SHA256withRSA
Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [ 
  KeyIdentifier [ 
    0000: BA F7 92 5B D1 5A 1C 29 40 8E 15 C7 73 63 A5 C8 ...
    0010: A8 7E 4D 17
  ]
]
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol" maxThreads="150" SSLEnabled="true" scheme="https" secure="true" clientAuth="false" sslProtocol="TLS" />
Your connection is not private

Attackers might be trying to steal your information from 192.168.99.101 (for example, passwords, messages, or credit cards). NET::ERR_CERT_AUTHORITY_INVALID
Apache Tomcat/8.0.33

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
- Security Considerations HOW-TO
- Manager Application HOW-TO
- Clustering/Session Replication HOW-TO

Developer Quick Start
- Tomcat Setup
- First Web Application
- Realms & AAA
- JDBC DataSources
- Examples
- Servlet Specifications
- Tomcat Versions

Managing Tomcat
For security, access to the manager webapp is restricted.

Documentation
- Tomcat 8.0 Documentation
- Tomcat 8.0 Configuration

Getting Help
- FAQ and Mailing Lists
The following mailing lists are
Certificate Pinning
String alias = args[0];

try {
    KeyStore ks = KeyStore.getInstance("JKS");
    ks.load(new FileInputStream("ic-keystore"), "password".toCharArray());

    Certificate cert = ks.getCertificate(alias);
    if (cert == null) {
        System.out.println("Alias "+ alias + " not found.");
        return;
    }

    byte[] key =
        ((X509Certificate)cert).getPublicKey().getEncoded();
    MessageDigest md = MessageDigest.getInstance("SHA1");
    byte[] keyHash = md.digest(key);

    String khs = DatatypeConverter.printHexBinary(keyHash);
    System.out.println("Hash: " + khs);
}

catch (Exception e) {
    System.out.println("Exception: " + e);
}
public void verifyPin(String pinString, X509Certificate cert)
    throws CertificateException {
    try {
        byte[] key = cert.getPublicKey().getEncoded();
        MessageDigest md = MessageDigest.getInstance("SHA1");
        byte[] keyHash = md.digest(key);
        byte[] pin = DatatypeConverter.parseHexBinary(pinString);

        if (!Arrays.equals(keyHash, pin)) {
            System.out.println("Pin mismatch. Server hash: " +
                    DatatypeConverter.printHexBinary(keyHash));
            throw new CertificateException("Pin doesn't match.");
        }
    }
    catch (java.security.NoSuchAlgorithmException nsa) {
        System.out.println("NoSuchAlgorithm.");
    }
    catch (CertificateException ce) {
        throw ce;
    }
}
Resources
Diagnosing TLS, SSL, and HTTPS

By Erik Costlow-Oracle on Jul 02, 2014

When building inter-connected applications, developers frequently interact with TLS-enabled protocols like HTTPS. With recent emphasis on encrypted communications, I will cover the way in which the JDK evolves regarding protocols, algorithms, and changes, as well as some advanced diagnostics to better understand TLS connections like HTTPS.

Most developers will not have to do this level of diagnosis in the process of writing or running applications. In the event that you do, the following information should provide enough information to understand what’s happening within secure connections.

Stability: The evolution of protocols and algorithms

For the last 15 years (since 1998), the Java platform has evolved through the Java Community Process where companies, organizations, and dedicated individuals develop and vote on specifications to determine what makes up the Java Platform. Much of the efforts are centered on compatibility, like the TCK, ensuring that different implementations are compatible with each-other and that developers can predict how their applications will run. We are not changing critical default options (like TLS protocol) within minor versions.

The following chart depicts the protocols and algorithms supported in each JDK version:

<table>
<thead>
<tr>
<th>TLS Protocols</th>
<th>JDK 8 (March 2014 to present)</th>
<th>JDK 7 (July 2011 to present)</th>
<th>JDK 6 (2006 to end of public updates 2013)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLSv1.2</td>
<td>TSLv1.2</td>
<td>TSLv1.1</td>
<td>TSLv1.1 (JDK 6 update 111 and above)</td>
</tr>
<tr>
<td>TLSv1.1</td>
<td>TSLv1.1</td>
<td>TSLv1.1</td>
<td>TSLv1.1 (default)</td>
</tr>
<tr>
<td>SSLv3</td>
<td>SSLv3</td>
<td>SSLv3</td>
<td>SSLv3</td>
</tr>
</tbody>
</table>

JSSE Ciphers:

- Cipher in JDK 8
- Cipher in JDK 7
- Cipher in JDK 6

Reference:

- JDK 8 JSSE
- JDK 7 JSSE
- JDK 6 JSSE

Java Cryptography Extension, Unlimited Strength (explained later):

- JCE for JDK 8
- JCE for JDK 7
- JCE for JDK 6

Sample Java code for making an HTTPS connection

Making an HTTPS connection in Java is relatively straight-forward. I will post the code here with the intent focused on tuning and understanding the underlying capabilities.

Sample back-end code for making an SSL connection:

```java
final url = new URL("https://example.com");
```
The Most Common Java Keytool Keystore Commands

Java Keytool is a key and certificate management utility. It allows users to manage their own public/private key pairs and certificates. It also allows users to cache certificates. Java Keytool stores the keys and certificates in what is called a keystore. By default the Java keystore is implemented as a file. It protects private keys with a password. A Keytool keystore contains the private key and any certificates necessary to complete a chain of trust and establish the trustworthiness of the primary certificate.

Each certificate in a Java keystore is associated with a unique alias. When creating a Java keystore you will first create the .jks file that will initially only contain the private key. You will then generate a CSR and have a certificate generated from it. Then you will import the certificate to the keystore including any root certificates. Java Keytool also several other functions that allow you to view the details of a certificate or list the certificates contained in a keystore or export a certificate.

Note: For easier management of your Java Keystores (using a GUI) check out Portecle. If you need to buy a certificate, try to compare SSL with our SSL Wizard.

Below, we have listed the most common Java Keytool keystore commands and their usage:

Java Keytool Commands for Creating and Importing

These commands allow you to generate a new Java Keytool keystore file, create a CSR, and import certificates. Any root or intermediate certificates will need to be imported before importing the primary certificate for your domain.

- Generate a Java keystore and key pair
  keytool -genkey -alias mydomain -keyalg RSA -keystore keystore.jks -keysize 2048

- Generate a certificate signing request (CSR) for an existing Java keystore
  keytool -certreq -alias mydomain -keystore keystore.jks -file mydomain.csr

- Import a root or intermediate CA certificate to an existing Java keystore
  keytool -import -trustcacerts -alias root -file Thawte.crt -keystore keystore.jks

- Import a signed primary certificate to an existing Java keystore
  keytool -import -trustcacerts -alias mydomain -file mydomain.crt -keystore keystore.jks

- Generate a keystore and self-signed certificate (see How to Create a Self Signed Certificate using Java Keytool for more info)
  keytool -genkey -keyalg RSA -alias selfsigned -keystore keystore.jks -storepass password -validity 360 -keysize 2048
OWASP Cheat Sheets

- Authentication
- Choosing and Using Security Questions
- Clickjacking Defence
- Cross-Site Request Forgery (CSRF) Prevention
- Cryptography Storage
- DOM based XSS Prevention
- Forgot Password
- HTML 5 Security
- Input Validation
- JAAS
- Logging
- Password Storage
- Pinning
- Query Parameterization
- REST Security
- Session Management
- SQL Injection Prevention
- Transport Layer Protection
- Unvalidated Redirects and Forwards
- User Privacy Protection
- Web Service Security
- XSS (Cross Site Scripting) Prevention

https://www.owasp.org/index.php/Cheat_Sheets
OWASP User Groups

The Open Web Application Security Project (OWASP) is a 501(c)(3) worldwide not-for-profit charitable organization focused on improving the security of software. Our mission is to make software security visible, so that individuals and organizations worldwide can make informed decisions about true software security risks.

Welcome, AppSec Professionals!

Quarterly Seminar

Improving Enterprises
One Easton Ova, Suite 175, Columbus, OH (map)

For our first Lunch Seminar in a while, we are especially honored to bring in Matthew Curtin. Title: Crypto War II: Protecting the Infrastructure Abstract: During the... LEARN MORE

Hosted by: Bill S. (Organizer)

Thu May 28
11:00 AM
RSVP
9 going
0 comments

https://www.owasp.org/index.php/OWASP_Chapter