SSL Testing Tool

by Siddhartha De MVB · Nov. 02, 17 · Security Zone · Tutorial

If you have a large number of servers, which are configured with SSL/TLS and you are out of track on their certificate validity, now all of sudden you are worried if some of the certificates are expired.

Or if I think in some other scenario where you are required to understand underlying SSL/TLS configuration of your servers e.g. CipherSuits, Protocols, etc.

Yes, in the traditional way, you can get all the information of your SSL/TLS configuration by logging into an individual server and checking the certificates, but it is very difficult if your environment size is very large.

To overcome this problem, I have to build a tool, which will give you get all the required details.

Source Code:

```
import java.io.FileInputStream;
import java.math.BigInteger;
import java.security.KeyStore;
import javax.net.ssl.KeyManager;
import javax.net.ssl.KeyManagerFactory;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.SSLSocket;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.TrustManager;
import javax.net.ssl.TrustManagerFactory;
import javax.security.cert.X509Certificate;

/**
*
* @author sidd
**/

public class SSLFactory_Client {
    public static void main(String[] args){
       String hostname;
       Integer port;
```

```
        Integer port;
        if(args.length!=2){
            hostname = "google.com";
            port = 443;
        }else{
            hostname = args[0];
            port = Integer.valueOf( args[1]);
        }

        SSLFactory_Client sclient = new SSLFactory_Client();
        SSLContext sslContext = sclient.createSSLContext();
        try {
            SSLSocketFactory sslSocketFactory = sslContext.getSocketFactory();
            SSLSocket sslSocket = (SSLSocket) sslSocketFactory.createSocket(hostname, port);
            sslSocket.startHandshake();
            SSLSession sslSession = (SSLSession) sslSocket.getSession();

            System.out.println("SSLSession :");
            System.out.println("\tSessionID: "+  new BigInteger(sslSession.getId()));
            System.out.println("\tProtocol : "+sslSession.getProtocol());
            System.out.println("\tCipher suite : "+sslSession.getCipherSuite());
            System.out.println("\tServer: "+sslSession.getPeerHost());
            System.out.println("\tSSL Port: "+sslSession.getPeerPort());

            System.out.println("\nSupported Protocol :");
            for(int i=0;i<sslSocket.getEnabledProtocols().length;i++){
                System.out.println("\t"+sslSocket.getEnabledProtocols()[i]);
            }

            System.out.println("\nSupported CipherSuites: ");
            for(int j=0;j<sslSocket.getEnabledCipherSuites().length;j++){
                System.out.println("\t"+sslSocket.getEnabledCipherSuites()[j]);
            }

            X509Certificate[] certs = (X509Certificate[]) sslSession.getPeerCertificateChain();
            System.out.println("\nCertificate Chain Info :");
            for (int i =0;i<certs.length;i++){
                System.out.println("\tSubject DN :"+((X509Certificate) certs[i]).getSubjectDN());
                System.out.println("\tIssuer DN  : "+((X509Certificate) certs[i]).getIssuerDN());
                System.out.println("\tSerial No. : "+((X509Certificate) certs[i]).getSerialNumber(
                System.out.println("\tExpires On : "+((X509Certificate) certs[i]).getNotAfter()+"\n
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    private SSLContext createSSLContext(){
        try{
            KeyStore keyStore = KeyStore.getInstance("JKS");
            keyStore.load(new FileInputStream("/opt/jdk1.8.0_102/jre/lib/security/cacerts"),"change

            // Create key manager
            KeyManagerFactory keyManagerFactory = KeyManagerFactory.getInstance("SunX509");
            keyManagerFactory.init(keyStore, "changeit".toCharArray());
            KeyManager[] km = keyManagerFactory.getKeyManagers();

            // Create trust manager
            TrustManagerFactory trustManagerFactory = TrustManagerFactory.getInstance("SunX509");
```

```
            TrustManagerFactory trustManagerFactory = TrustManagerFactory.getInstance("SunX509");
            trustManagerFactory.init(keyStore);
            TrustManager[] tm = trustManagerFactory.getTrustManagers();

            // Initialize SSLContext
            SSLContext sslContext = SSLContext.getInstance("TLSv1.2");
            sslContext.init(km,  tm, null);
            return sslContext;
        } catch (Exception ex){
            ex.printStackTrace();
            return null;
        }
    }
}
```

Compile the code using `javac` (e.g. javac SSLFactory_Client .java).

Now, you can execute the program. You need to pass the hostname and port during the execution (e.g java SSLFactory_Client "google.com" 443) and you will get the output, which should look something like the screenshot below.

Output:

```
SSLSession :
        SessionID: -207907921674146617136738714528623010318863302418017493929237052518063435119 05
        Protocol : TLSv1.2
        Cipher suite : TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
        Server: redhat.com
        SSL Port: 443

Supported Protocol :
        TLSv1
        TLSv1.1
        TLSv1.2

Supported CipherSuites:
        TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
        TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
        TLS_RSA_WITH_AES_128_CBC_SHA256
        TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
        TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
        TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
        TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
        TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
        TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
        TLS_RSA_WITH_AES_128_CBC_SHA
        TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
        TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
        TLS_DHE_RSA_WITH_AES_128_CBC_SHA
        TLS_DHE_DSS_WITH_AES_128_CBC_SHA
        TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
        TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
        TLS_RSA_WITH_AES_128_GCM_SHA256
        TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256
        TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256
        TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
        TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
        TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
        TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
        SSL_RSA_WITH_3DES_EDE_CBC_SHA
        TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
        TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA
        SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
        SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
        TLS_EMPTY_RENEGOTIATION_INFO_SCSV

Certificate Chain Info :
        Subject DN :CN=*.redhat.com, O=Red Hat Inc., L=Raleigh, ST=North Carolina, C=US
        Issuer DN  : CN=DigiCert SHA2 High Assurance Server CA, OU=www.digicert.com, O=DigiCert Inc, C=US
        Serial No. : 27205353483955024814274626922053548 81
        Expires On : Fri Jul 19 17:30:00 IST 2019
```

```
Subject DN :CN=DigiCert SHA2 High Assurance Server CA, OU=www.digicert.com, O=DigiCert Inc, C=US
Issuer DN  : CN=DigiCert High Assurance EV Root CA, OU=www.digicert.com, O=DigiCert Inc, C=US
Serial No. : 64898770745461662225103809517619173 43
Expires On : Sun Oct 22 17:30:00 IST 2028
```

Note: This program can also be used for testing two-way SSL/TLS connections.

Topics: SECURITY, SERVER SECURITY, SSL, TLS

Published at DZone with permission of Siddhartha De , DZone MVB. See the original article here.