SSL 101 in Java

Java Workshop V.2.0 supplemental material for Week #1

(author: Mikalai Zaikin mzaikin@ibagroup.eu)

Когда вы из Java SE устанавливаете HTTPS соединение к удалённому серверу, и вы не используете например SSL mutual authentication (как мы в приложении) то вам не нужно генерировать свои private / public SSL ключи. Это логично, так как пользуясь браузером для подключения к сайтам вы сначала не генерируете приватный и паблик ключи.

Для асимметричного шифрования будут использоваться ключи с сервера (ассиметричное шифрование используется в начале сессии для обмена симметричным ключом).

SIDENOTE: Также вы должны знать, что паблик ключ сервера (обернутый в сертификат) можно скачать явным образом (либо утилитой либо даже браузером) и сохранить в файл на файловой системе.

Для таких случаев, когда мы подключаемся к внешнему HTTPS серверу, чтобы удостовериться в защищённости шифрования, в JDK уже есть предустановленный truststore файл с сертификатами известных основных Certificate Authorities (CA).

Этот файл называется **cacerts** и находится в папке (например)

```
C:\Program Files\Eclipse Adoptium\jdk-17.0.2.8-hotspot\lib\security\
```

Вы можете посмотреть какие сертификаты каких CA предустановлены (исторически пароль по умолчанию всегда **changeit**) в вашем JDK:

```
keytool -list -v -keystore cacerts
```

Полная команда

```
C:\Program Files\Eclipse
Adoptium\jdk-17.0.2.8-hotspot\lib\security>keytool -list -v -keystore
cacerts  > c:\temp\java.17.certs.txt
```

Что же мы видим в файле

```
Keystore type: JKS
Keystore provider: SUN

Your keystore contains 131 entries
```

```
Alias name: subject=c = at, o = e-commerce monitoring gmbh, cn =
globaltrust 2020
Creation date: Jan 20, 2022
Entry type: trustedCertEntry

Owner: CN=GLOBALTRUST 2020, O=e-commerce monitoring GmbH, C=AT
Issuer: CN=GLOBALTRUST 2020, O=e-commerce monitoring GmbH, C=AT
Serial number: 5a4bbd5afb4f8a5bfa65e5
Valid from: Mon Feb 10 02:00:00 EET 2020 until: Sun Jun 10 03:00:00
EEST 2040
Certificate fingerprints:
      SHA1:
D0:67:C1:13:51:01:0C:AA:D0:C7:6A:65:37:31:16:26:4F:53:71:A2
      SHA256:
9A:29:6A:51:82:D1:D4:51:A2:E3:7F:43:9B:74:DA:AF:A2:67:52:33:29:F9:0F:
9A:0D:20:07:C3:34:E2:3C:9A
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 4096-bit RSA key
Version: 3
```

1) 131 сертификат (реально Certificate Authorities существует меньше, так как одному CA могут принадлежать несколько сертификатов).

2) Признаком сертификата является **`trustedCertEntry`**

В Java разделение truststore / keystore – условное и логическое, один и тот же файл может играть обе роли, но вот по тексту **`trustedCertEntry`** вы не ошибетесь что тут хранится сертификат. Если бы это был приватный ключ то энтри выглядела бы как **`Entry type: PrivateKeyEntry`**

Повторюсь – в JDK out-of-box есть только truststore, так как keystore используется для приватных ключей, которые хранят ваши атрибуты идентифицирующие вашу организацию. Вы сами можете генерировать ключи и создавать создать keystore, да и truststore тоже рекомендуется свой завести. Мы рассматриваем что есть по умолчанию.

Пробуем создать защищённое соединение с внешним HTTPS сервером, используя встроенный в JDK truststore (он будет использоваться так как мы не указали своего личного).

Наш тестовый класс

```
import javax.net.ssl.SSLSocket;
import javax.net.ssl.SSLSocketFactory;
```

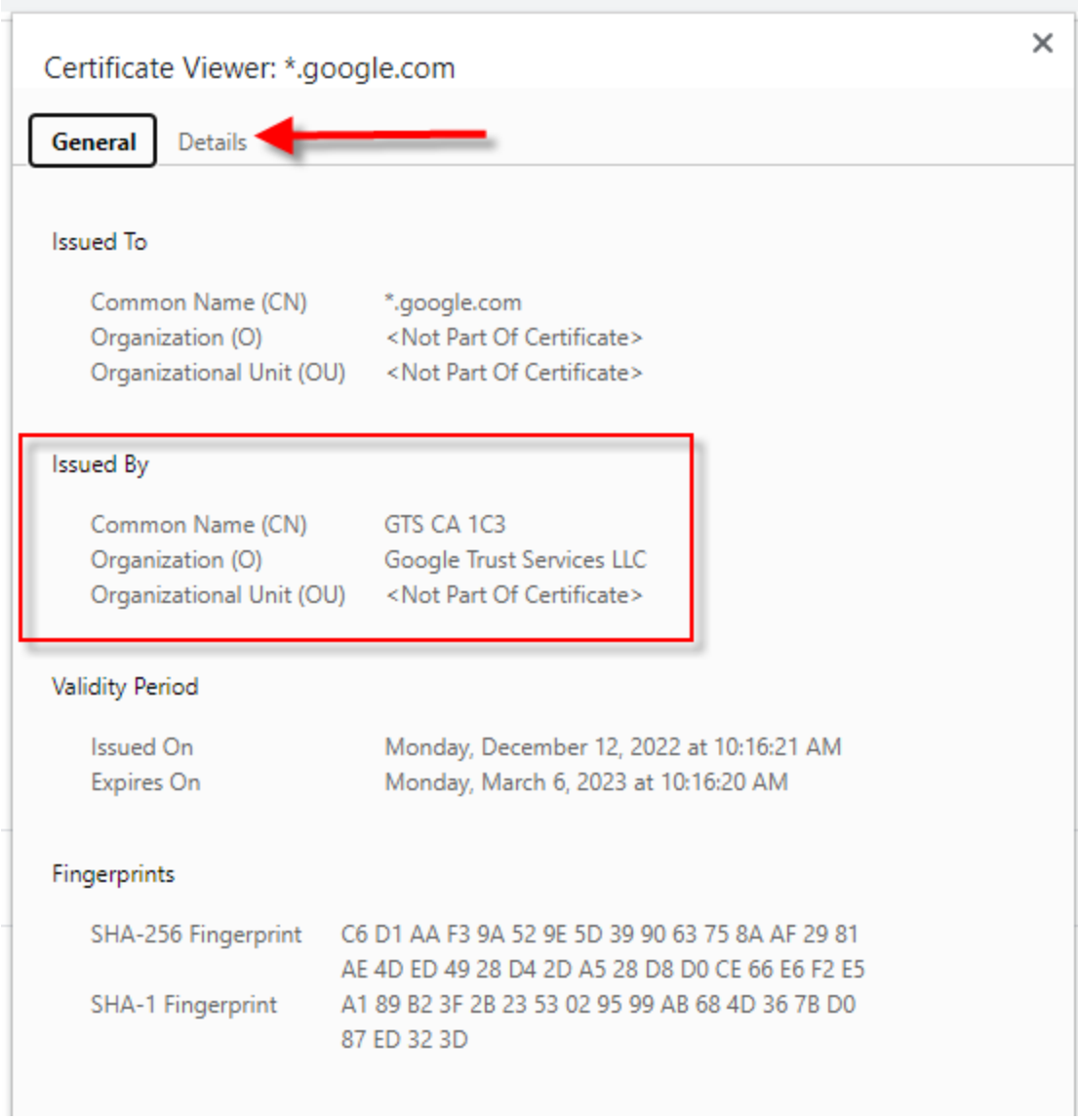
```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

public class HTTPSClient {
    public static void main(String[] args) throws IOException {
        String host = "google.com";
        Integer port = 443;
        SSLSocketFactory sslsocketfactory = (SSLSocketFactory)
SSLSocketFactory.getDefault();
        SSLSocket sslsocket = (SSLSocket)
sslsocketfactory.createSocket(host, port);
        InputStream in = sslsocket.getInputStream();
        OutputStream out = sslsocket.getOutputStream();
        out.write(1);
        while (in.available() > 0) {
            System.out.print(in.read());
        }
        System.out.printf("Secured connection to %s:%d established
successfully", host, port);
    }
}
```

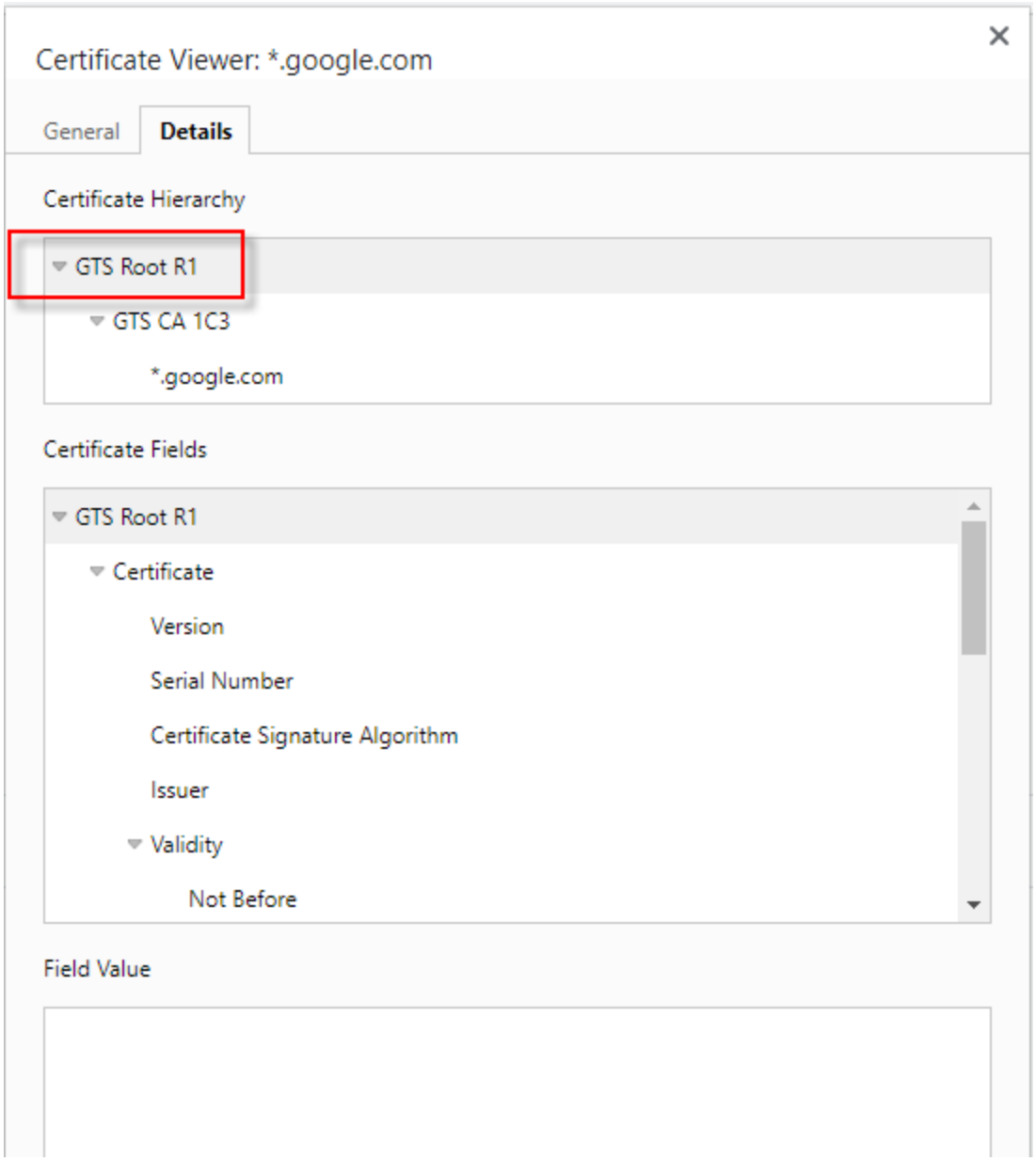
Output:

```
Secured connection to google.com:443 established successfully
Process finished with exit code 0
```

Как произошла магия? Открываем браузером https://google.com, жмем на замочек, смотрим сертификат



Подписан корневым



Идём в файл с дампом сертификатов и находим **GTS Root R1**

```
Alias name: subject=c = us, o = google trust services llc, cn = gts
root r1
Creation date: Jan 20, 2022
Entry type: trustedCertEntry

Owner: CN=GTS Root R1, O=Google Trust Services LLC, C=US
Issuer: CN=GTS Root R1, O=Google Trust Services LLC, C=US
Serial number: 6e47a9c54b470c0dec33d089b91cf4e1
```

```
Valid from: Wed Jun 22 03:00:00 EEST 2016 until: Sun Jun 22 03:00:00
EEST 2036
Certificate fingerprints:
      SHA1:
E1:C9:50:E6:EF:22:F8:4C:56:45:72:8B:92:20:60:D7:D5:A7:A3:E8
      SHA256:
2A:57:54:71:E3:13:40:BC:21:58:1C:BD:2C:F1:3E:15:84:63:20:3E:CE:94:BC:
F9:D3:CC:19:6B:F0:9A:54:72
Signature algorithm name: SHA384withRSA
Subject Public Key Algorithm: 4096-bit RSA key
Version: 3
```

Теперь, для теста возьмем HTTPS веб сайт с самоподписанным (self-signed) сертификатом, которого гарантированно нет в нашем системном truststore

Хороший вебсайт для таких тестов https://badssl.com/

Конкретно наш кейс https://self-signed.badssl.com/

Тестируем, используя тот же класс, только новый хост

```
String host = "self-signed.badssl.com";
Integer port = 443;
```

Результат

```
Exception in thread "main" javax.net.ssl.SSLHandshakeException: PKIX
path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to
find valid certification path to requested target
     at
java.base/sun.security.ssl.Alert.createSSLException(Alert.java:131)
     at
java.base/sun.security.ssl.TransportContext.fatal(TransportContext.ja
va:371)
```

Такое можно исправить, добавив руками сертификат сервера в truststore (системный, но лучше пользовательский). Для этого и пригодится возможность скачивать сертификат с удалённого сервера в локальный файл и импорта с файловой системы.

Другой вариант исправления – отключить программно проверку сертификатов, т.е. шифрование будет работать, но принадлежность сертификата доверенной стороне не будет выполняться. Такой способ снижает безопасность приложения, но иногда он допустим (например, между микро сервисами внутри интранета).

Но мы этого делать не будем.

Как настроить truststore – изучаем на неделе №1 в рамках IBA Java Workshop 2.0

;-)