OpenSSL을 이용한 https 설정

OpenSSL + Tomcat

대덕인재개발원 : 최희연

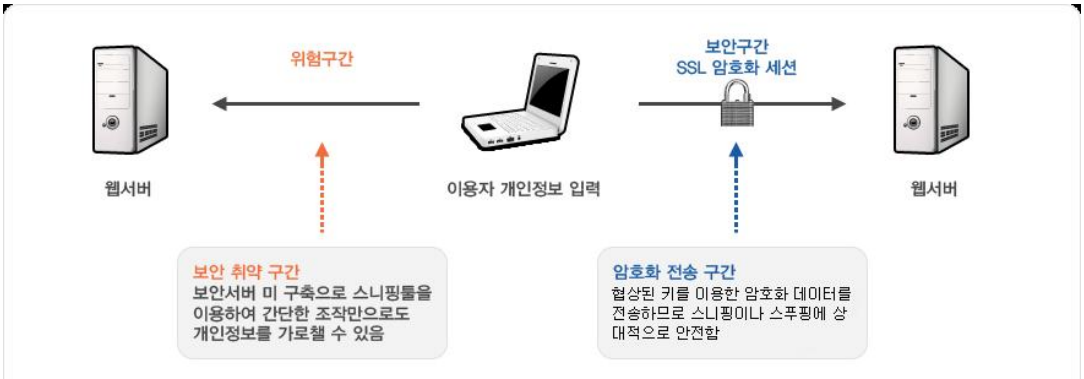
❑ SSL/TLS

- SSL(Secure Socket Layer) : C/S 환경에서 TCP 기반의 어플리케이션에 종단간 보안 서비스를 제공하기 위한 전송계층(Transport Layer) 보안 프로토콜로 전송계층과 어플리케이션 계층(Application layer) 사이에서 안전한 채널을 형성하는 역할을 하며, 각 어플리케이션 프로토콜이 SSL을 이용하는 경우, 기본 프로토콜과 구분하기 위해 다음과 같은 well-known 포트를 할당받는다.
  - http(80) - http over ssl, https(443)
  - ftp(21) - ftp over ssl, ftps(990)
  - smtp(25) - smtp over ssl, smtps (465)
  - Imap4(143) – imap4 over ssl (993)
  - Pop3(110) - pop3 over ssl (995)

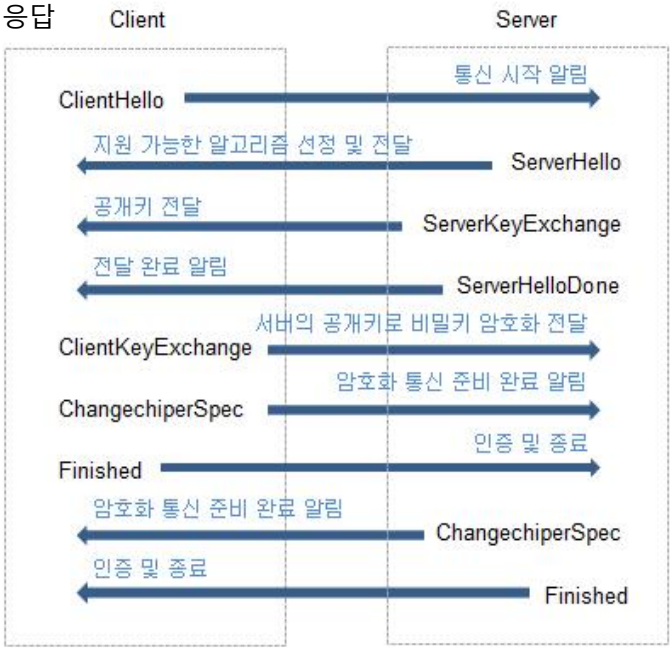
<table>
<tr><td>SSL handshake protocol</td><td>SSL cipher change protocol</td><td>SSL alert protocol</td><td>Application Protocol (eg. HTTP)</td></tr>
<tr><td colspan="4">SSL Record Protocol</td></tr>
<tr><td colspan="4">TCP</td></tr>
<tr><td colspan="4">IP</td></tr>
</table>

- SSL은 어플리케이션 프로토콜에 다음과 같은 보안 특성을 보장한다.
  - 기밀성(Confidentiality) : SSL을 이용한 보안 채널을 통해 전송되는 데이터는 종단간에 협상된 암호화 방식에 따라 암호화 되어 전송되므로 비공개 전송을 보장한다(대칭키 방식).
  - 무결성(Integrity) : HMAC(Hash MAC) 등의 메시지 인증 코드 알고리즘을 이용해 전송 메시지의 위변조 여부를 확인할 수 있도록 하여 전송 데이터의 무결성을 보장한다(단방향 해시 알고리즘).
  - 인증(Authentication) : 공개키 인증서를 이용한 서버/클라이언트간 상호 인증을 수행한다(비대칭키 방식).
- 위와 같은 보안 특성을 갖는 SSL 이 세부적으로 제공하는 보안 기능은 다음과 같다.
  - 상호 인증 : 공개키 인증서를 이용한 클라이언트/서버의 상호 인증 (CA 기반의 인증 체계)
  - 메시지 압축 : 기본 구조는 무압축이나, 협상을 통해 압축 알고리즘 변경 가능
  - 메시지 인증 : 메시지 인증 코드(MAC)에 의한 메시지 무결성 제공 : HMAC
  - 암호화용 세션 키 생성을 위한 키 교환 : RSA, Diffie-Hellman
  - 생성된 공유 비밀키에 의해 암호화된 종단간 안전한 연결 통로 제공 : RC4, IDEA, DES, 3DES 등

  즉, SSL 이란 종단간 연결수립 시 handshake 단계에서 보안 채널에서 사용될 주요 알고리즘들이 결정되면, 종단간의 데이터 전송 과정에서 결정된 암호방식에 따라 공유된 암호키를 이용하여 암호화된 데이터를 전송함으로 보안을 유지하는 프로토콜의 개념이라 할 수 있다. 예를 들어 HTTPS 란 SSL 위에서 동작하는 HTTP 프로토콜의 의미를 가지며, HTTPS 를 이용한 데이터 전송시 악의적 도감청이나 데이터 변조의 위험성을 낮출 수 있게 되는 것이다.



- SSL 은 Certificate Authority(CA) 라 불리는 제 3 자로부터 서버와 클라이언트를 인증받는 구조로 되어 있로, SSL 인증서가 이 프로토콜에서 가장 핵심이 되는 요소이며 클라이언트와 서버간의 통신을 제3자가 보증해주는 전자화된 문서의 개념이다. 이 인증서를 이용한 통신 과정은 다음과 같다.
  1. 클라이언트가 서버에 접속하면 서버는 클라이언트에게 인증서를 전달한다.
  2. 클라이언트는 인증서에 대한 신뢰 검증 후 인증서에 포함된 서버의 공개키를 추출한다.
  3. 클라이언트가 세션키로 사용할 임의의 메시지를 서버의 공개키로 암호화하여 서버에 전송한다.
  4. 서버는 자신의 개인키로 세션키를 복호화하여 그 키를 사용해 대칭키 암호방식으로 메시지를 암호화하여 클라이언트와 통신하게 된다.
- 클라이언트로 전송된 SSL 디지털 인증서에는 서비스 정보(인증서 발급기관, 인증 서비스 정보)와 서버 측에서 제공한 공개키가 포함되며, 클라이언트로 전송된 인증서에 포함된 공개키는 메시지 전송에 사용될 비밀키를 교환하는 과정에서 비밀키 암호화에 사용된다.
- 위의 과정을 지원하기 위한 SSL 프로토콜의 동작구조를 정리하면 다음과 같다.
  1. Handshake(완전협상) : 데이터 전송시 사용될 암호화 방식이나 메시지 압축 방식, 메시지 인증 코드 생성 방식 등이 결정된다.
     1) Client Hello : 클라이언트의 서버 접속
        ① 서버로부터 취득한 인증서의 CA 정보를 클라이언트가 보유한 CA 리스트에서 검색 : 클라이언트에 설치된 CA 인증서에 포함된 CA의 공개키로 서버의 인증서를 복호화한다.
        ② 클라이언트 랜덤 생성 : master secret 생성시 salt 역할
        premaster secret + salt => master secret + salt => key block
        ③ 클라이언트에서 활용 가능한 암호화 방식을 서버로 전송(Cipher suite)
     2) Server Hello : 클라이언트 접속에 대한 응답
        ① 클라이언트가 보낸 Cipher suite 내에서 활용 가능한 암호화 방식에 대한 정보를 클라이언트로 전송
        ② 서버의 랜덤 생성
     3) Server Key Exchange : 서버의 공개키를 클라이언트로 전송
     4) Client Key Exchange :
        Premaster secret(난수) 를 생성하여 서버의 공개키로 암호화한 후 전송.
        서버는 Premaster secret + salt(랜덤) -> master secret + salt -> 비밀키로 사용함.
     5) Change cipher spec : 협상한 암호명세의 적용 알림
     6) Server Hello Done



  2. 단축협상 : 완전협상에 의해 결정된 암호화 방식에 따라 암호화에 사용될 비밀키를 교환하는데, 이 단계에서 master secret 과 랜덤을 조합한 키 블록을 생성하고 CA로부터 인증된 서버의 인증서에 포함된 서버의 공개키로 암호화하여 교환한다. 데이터 전송시마다 이 단계부터 반복되며, 사용된 키블록(세션키)는 전송 완료 후 폐기된다.
  3. 데이터 전송 : 협상단계에서 결정된 압축방식으로 메시지를 압축하고, MAC 알고리즘에 따라 MAC 값을 추가한 후, 비밀키를 이용하여 암호화된 데이터를 전송한다.

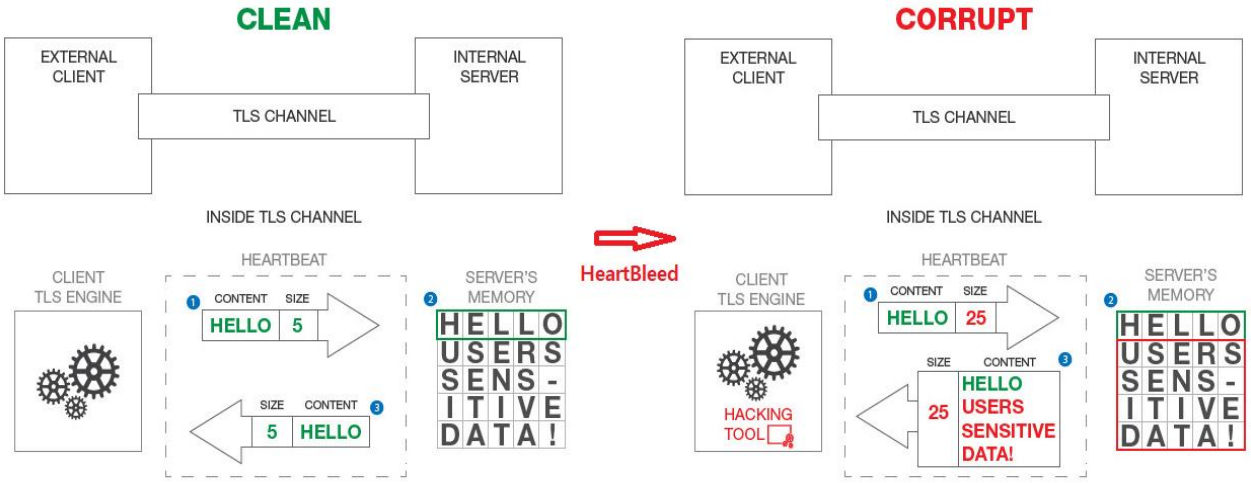
❑ 인증서 발급

- SSL 적용 서비스를 위해 공인된 CA 로부터 인증서가 필요한데, 여기에는 일정 비용과 절차가 필요하다. 인증서를 발급받는 방법에 따라 다음과 같은 종류의 인증서들이 사용될 수 있다.
  - 유료 인증 기관을 이용한 인증서 발급 : 공인된 CA(Comodo, DigitCert, GlobalSign)에 비용을 지불하고 받은 인증서 모듈(.cnf) 과 CA 인증 파일 등을 서버에 설치하여 인증서를 사용한다.
  - 무료 인증 기관을 이용한 인증서 발급 : 무료 CA(letsencrypt)에서 도메인 단위의 간단한 인증 절차를 통해 인증서를 이용할 수 있다. 이러한 경우, letsencrypt 는 각 시스템에 기본 설치된 루트 CA 들과의 cross singing 을 통해 서명한 인증서를 발급하게 된다.
  - 자가서명(Self-Signed) 사설 인증서 : OpenSSL에 CA 툴을 이용해 Root CA 를 생성하고, 이 CA 를 이용해 자가서명된 사설 인증서를 발급한 후 생성된 Root CA 를 클라이언트에 직접 설치한다.

❑ SSL Toolkit

SSL 툴에는 openSSL/boringSSL/libreSSL 등이 있으며, 다수의 어플리케이션에서 openSSL 이 많이 사용되는 편이나, openSSL 의 특정 버전에서 보안 취약점이 발견된바 있다.

- HeartBleed : Heartbeat 메시지 교환 중 발생하는 보안 취약점 (OpenSSL 1.0.1)
  - Heartbeat : openSSL 1.0.1 에서 추가된 확장모듈로 SSL이 매 연결시마다 재협상을 하지 않아도 연결을 유지하게 해주는 확장 규격을 의미한다. 클라이언트가 하트비트를 요청하면서 payload 와 payload 길이를 전송하면, 서버에서는 요청 사항을 확인하고, 요청의 payload 를 그대로 에코 응답으로 전송한다. 이 과정을 주기적으로 반복하여 클라이언트와 서버 사이의 연결을 유지한다.
  - HeartBleed : 하트비트 요청의 payload와 그 길이가 전송되면, 서버에서 payload 길이에 대해 검증하지 않고 에코 메시지를 반환하는 경우, 보안 정보가 유출될 수 있는 보안 취약점을 의미한다. 예를 들어, payload(Hello) + payload length(5) 의 하트비트 요청을 payload(Hello) + payload length(25) 와 같은 형태로 위조하여 전송하면, 서버에서는 25라는 길이에 대해 실제 payload 길이와 비교하지 않은 상태에서 에코메시지를 전송하기 때문에 서버의 메모리에서 저장되어있던 payload 와 메모리상 그 다음 주소에 포함된 비공개 데이터까지 위조된 payload 길이만큼 노출되는 상황이 발생하게 된다.

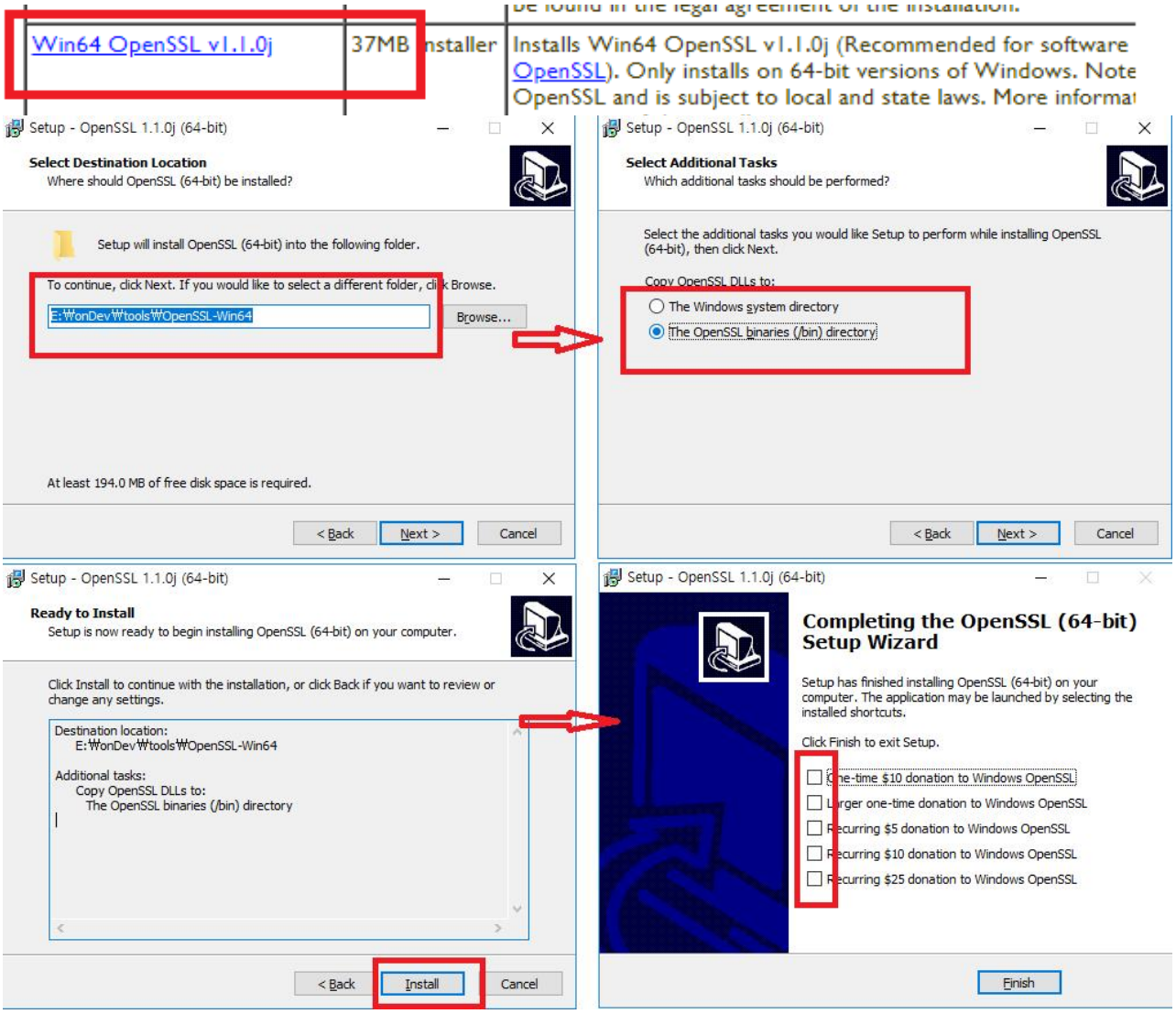


이러한 보안 취약점으로 인해 완전협상에서 결정된 cipher spec 이나 민감한 개인 정보등이 노출되는 문제가 발생할 수 도 있으며, 시스템의 버그를 이용했으나 정상적인 하트비트 메시지 처리 과정으로 기록되기 때문에 공격의 흔적이 남지 않는 다는 문제가 발생할 수 있다.

구글에서는 하트블리드 발생 이후, 해당 취약점에 대응하고 크롬, 안드로이드 환경에 최적화된 대안으로 BoringSSL 을 공개했으며, openBSD 에서는 LibreSSL 등의 대안 라이브러리가 주목받기도 했다. 다만, 하트블리드 취약점은 특정 openSSL 1.0.1에서 발견된 버그이므로, 1.0.1g 이후의 버전은 패치되어 취약점 노출이 없어, 버전 업그레이드 만으로도 하트블리드 취약점을 노린 해킹을 방어할 수 있다.

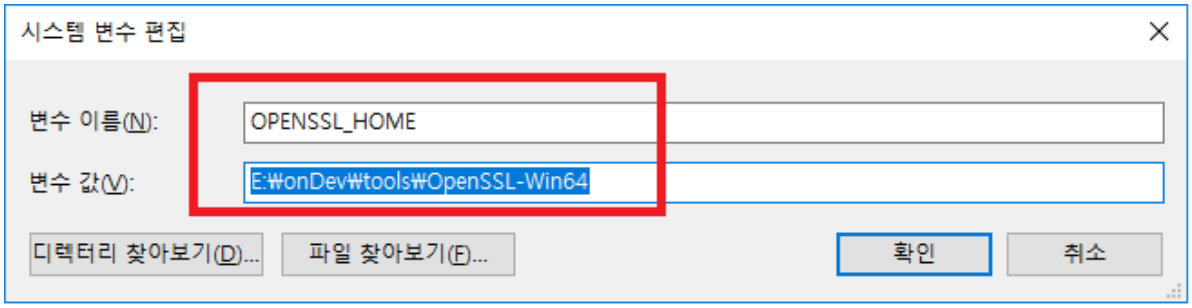
❑ OpenSSL 과 Tomcat 을 이용한 SSL 서버 설정

1. OpenSSL 설치(http://slproweb.com/products/Win32OpenSSL.html 윈도우 설치 파일 제공)



2. 환경변수 설정

   1) OPENSSL_HOME



   2) OPENSSL_CONF = %OPENSSL_HOME%₩bin₩openssl.cfg

   3) Path = 기존 path ;%OPENSSL_HOME%₩bin

3. Openssl.cnf 파일 생성

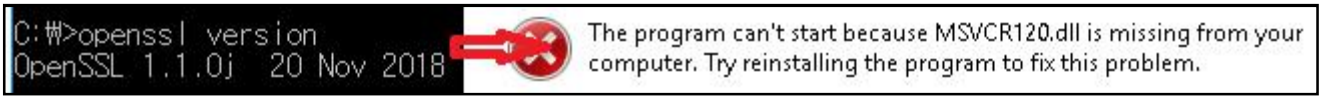
   %OPENSSL_HOME%₩bin₩openssl.cfg 파일을 복사하여 X509V3 설정 기본값을 가진 openssl.cnf 파일을 정의한다.

   1) X509 : 공개키 기반 구조(PKI)의 인증 표준으로 V3 까지 사용되고 있으며, 종단간 서로를 식별할 수 있는 공개키를 포함하고 있는 인증서를 발급하는 시스템 구조를 갖는다.

2) openssl.cnf 파일 수정

```
[ v3_req ]
# Extensions section name -extensions 옵션의 값으로 사용됨.
# Extensions to add to a certificate request
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName = @alt_names
[alt_names]
DNS.1 = localhost
DNS.2 = 도메인
IP.1 = 127.0.0.1
IP.2 = 외부IP
```

X509 Extensions 정보를 가진 설정파일로, 공개키를 포함한 X509 표준 인증서 파일(.crt) 생성시 사용할 수 있는 extensions 종류를 의미하는 섹션들이 정의되어 있다. ex) [v3_ca], [v3_req] 등.



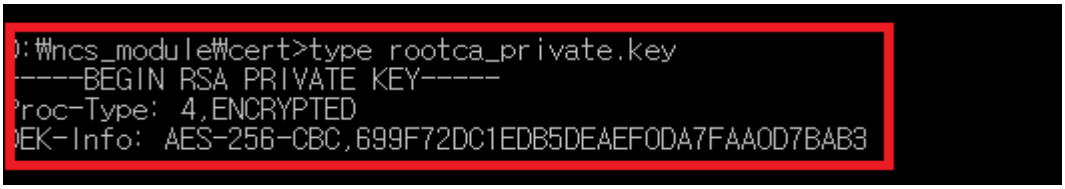
설치 후, 콘솔에서 openssl 명령 실행시 위와 같은 에러가 발생하는 경우, C++ 패키지 파일을 설치한다. (https://www.microsoft.com/ko-KR/download/details.aspx?id=40784)

4. Root CA 인증서 생성 : 이후 모든 작업은 관리자 권한으로 실행된 콘솔에서 작업한다.
개발 및 테스트 용도로 사용될 자가서명된 인증서(Self Signed Certificate)를 발급하기 위해, 사설 Root CA 를 생성한다. (https://www.openssl.org/docs/manmaster/man1/openssl.html)

1) 루트 인증기관의 개인키 생성(https://www.openssl.org/docs/manmaster/man1/genrsa.html)

RSA 개인키 생성 : aes256 방식으로 암호화된 2048비트 길이의 RSA 개인키를 생성한다.

개인키 생성에 사용된 password 는 향후 csr 파일과 crt 파일 생성해 사용되며, "-aes256" 옵션이 없는 경우, 비밀번호를 설정하지 않은 상태에서 개인키를 생성하게 된다.



2) CSR(Certificate Signing Requests) 파일 생성
(https://www.openssl.org/docs/manmaster/man1/req.html)
PKI 환경에서 CA에 인증서 발급 요청을 할때 사용되는 ASN.1 형식의 파일로, 사용자의 공개키와 암호화 알고리즘등을 포함하고 있다. 사설 CA 생성시 브라우저에 직접 설치할 CA 인증서 파일을 생성하기위해 csr 파일이 필요하다.

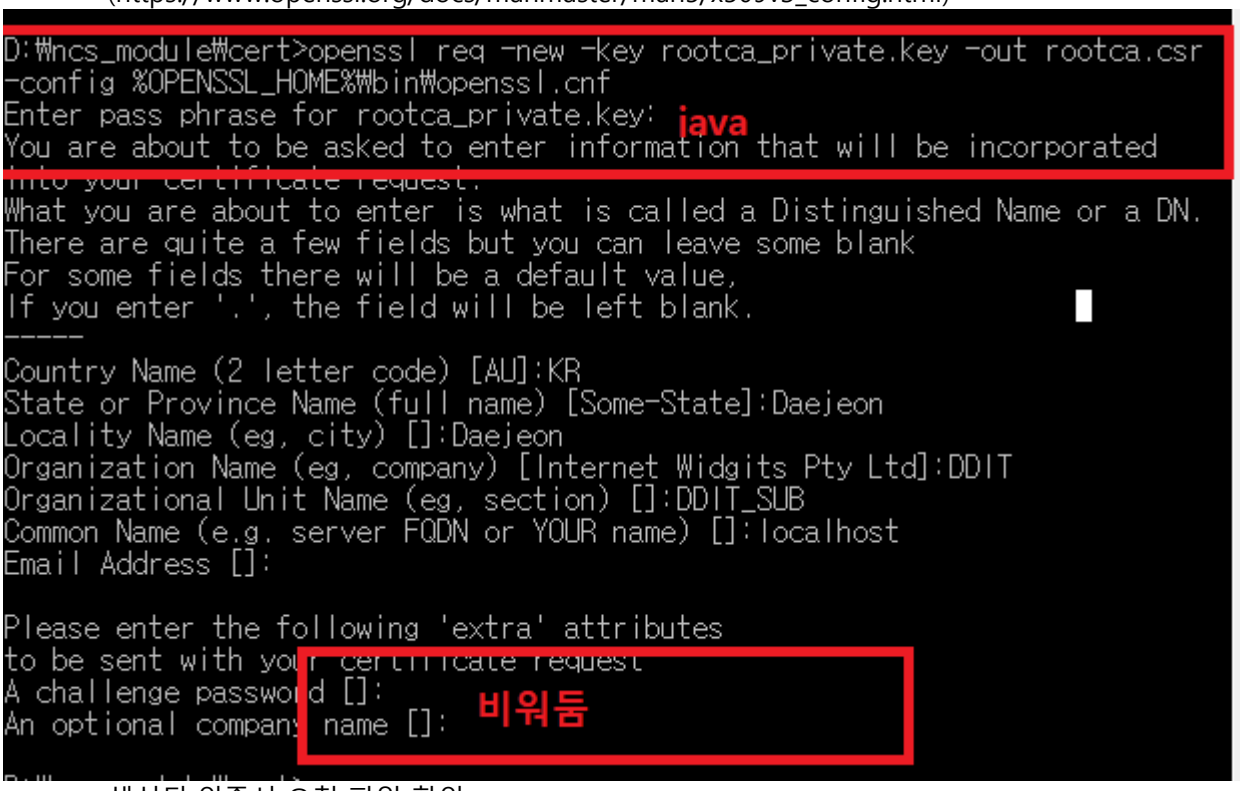
```
D:₩ncs_module₩cert>openssl req -new -key rootca_private.key -out rootca.csr
-config %OPENSSL_HOME%₩bin₩openssl.cnf
```

2) CSR(Certificate Signing Requests) 파일 생성(계속)

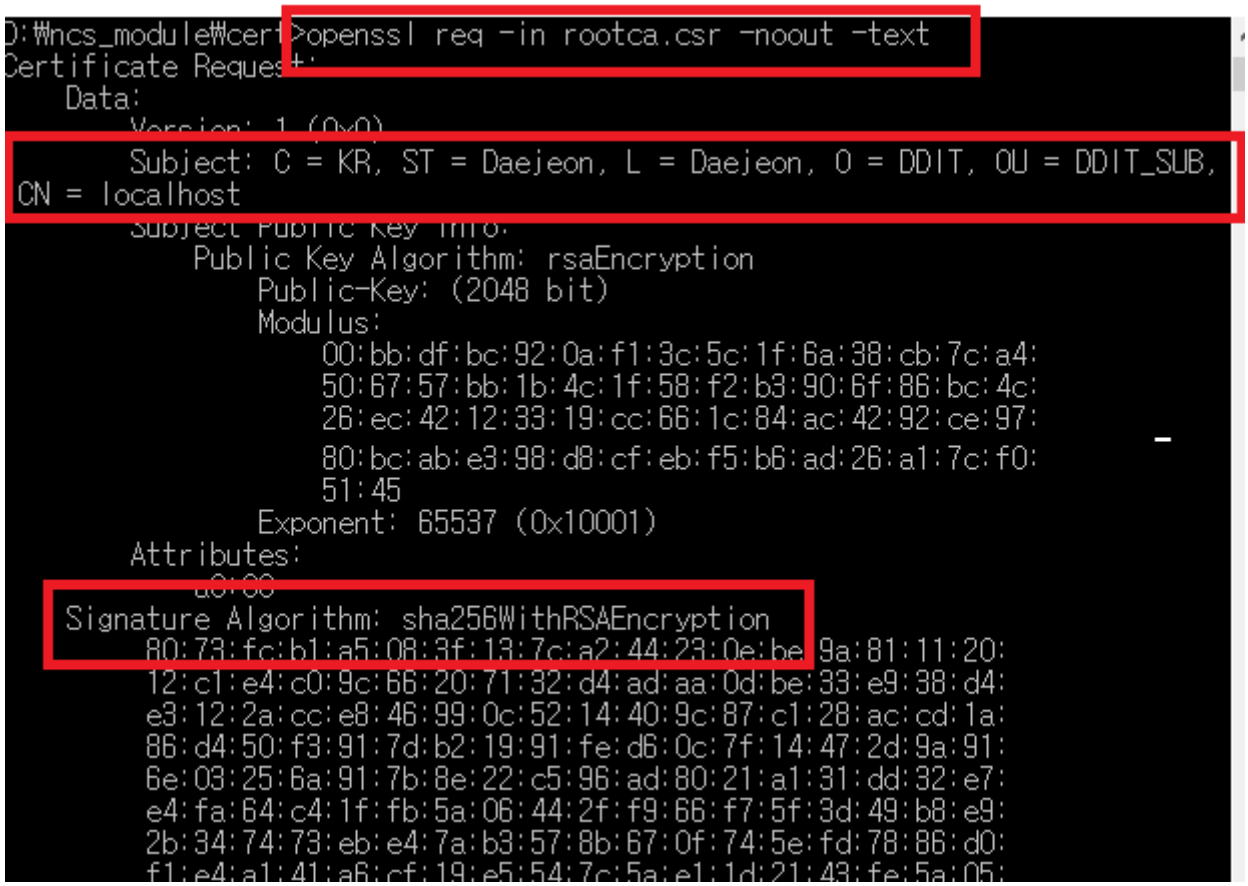
-key : 인증 요청자의 개인키 파일, -out : 생성할 인증 요청 파일의 이름,

-config : 인증서 발급에 필요한 설정들에 대한 정보를 가진 파일

(https://www.openssl.org/docs/manmaster/man5/x509v3_config.html)



생성된 인증서 요청 파일 확인



인증서 요청 파일의 발급자 정보와 해시 전자서명 알고리즘의 종류 등을 확인할 수 있다.

3) Root CA 인증서 생성 : 추후 클라이언트 브라우저에 직접 설치할 인증서 파일(CRT).
(https://www.openssl.org/docs/manmaster/man1/x509.html)

```
D:₩ncs_module₩cert>openssl x509 -req -days 3650 -set_serial 1 -in rootca.csr
 -signkey rootca_private.key -out rootca.crt -extensions v3_ca
Signature ok
subject=C = KR, ST = Daejeon, L = Daejeon, O = DDIT, OU = DDIT_SUB, CN = loc
alhost
Getting Private key
Enter pass phrase for rootca_private.key: java
```

-req : 인증서 요청 파일에 있는 설정을 바탕으로 인증서 생성
-days : 인증서 유효 기간 (10년)
-in : 인증서 기본 값을 가진 파일 ex) 인증서 요청 파일
-set_serial : 인증서의 시리얼 넘버, -signedkey 옵션과 함께 사용
-signkey : 자가 서명을 위해 필요한 자기 개인키
-out : 인증서 파일명(CRT 파일명)
-extensions : 인증서 설정파일(csr 파일 생성시 –config 설정값)의 extensions 섹션별로 구분된 설정 기본값을 선택 적용하기 위한 섹션명
(https://www.openssl.org/docs/manmaster/man5/x509v3_config.html)

```
D:₩ncs_module₩cert>openssl x509 -text -in rootca.crt
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 1 (0x1)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = KR, ST = Daejeon, L = Daejeon, O = DDIT, OU = DDIT_SUB,
CN = localhost
        Validity
            Not Before: May 14 06:06:50 2019 GMT
            Not After : May 11 06:06:50 2029 GMT
        Subject: C = KR, ST = Daejeon, L = Daejeon, O = DDIT, OU = DDIT_SUB,
CN = localhost
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:bb:df:bc:92:0a:f1:3c:5c:1f:6a:38:cb:7c:a4:
                    50:67:57:bb:1b:4c:1f:58:f2:b3:90:6f:86:bc:4c:
                    80:bc:ab:e3:98:d8:cf:eb:f5:b6:ad:26:a1:7c:f0:
                    51:45
                Exponent: 65537 (0x10001)
    Signature Algorithm: sha256WithRSAEncryption
         b0:5d:94:7f:bc:ce:e0:02:0f:22:33:15:1c:65:71:8f:c2:64:
```

인증서 발급자
self signed
인증 대상자

5. 자가 서명된 CA 로 발급할 로컬 서버(톰캣)의 인증서 생성

1) 로컬 서버용 개인키 생성 (https://www.openssl.org/docs/manmaster/man1/genrsa.html)

```
D:₩ncs_module₩cert>openssl genrsa -aes256 -out localhost_private.key 2048
Generating RSA private key, 2048 bit long modulus
.....................................+++++
.............+++++
e is 65537 (0x010001)
Enter pass phrase for localhost_private.key: java
Verifying - Enter pass phrase for localhost_private.key: java
```

2) 로컬 서버용 인증서 요청 파일 생성(https://www.openssl.org/docs/manmaster/man1/req.html)

```
D:₩ncs_module₩cert>openssl req -new -key localhost_private.key -out localhos
t.csr -config %OPENSSL_HOME%₩bin₩openssl.cnf
Enter pass phrase for localhost_private.key: java
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:KR
State or Province Name (full name) [Some-State]:Daejeon
Locality Name (eg, city) []:Daejeon
Organization Name (eg, company) [Internet Widgits Pty Ltd]:DDIT
Organizational Unit Name (eg, section) []:CLASS_207
Common Name (e.g. server FQDN or YOUR name) []:localhost
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

비워둠

3) 자가 서명된 Root CA 가 발급한 로컬 서버의 인증서 생성 : Root CA 를 신뢰할수있는 루트 인증기관으로 직접 설치된 브라우저에서 톰캣으로 요청 발생시 전송할 인증서.
(https://www.openssl.org/docs/manmaster/man1/x509.html)

```
D:₩ncs_module₩cert>openssl x509 -req -days 1825 -in localhost.csr -out local
host.crt -CA rootca.crt -CAcreateserial -CAkey rootca_private.key -extension
s v3_req -extfile %OPENSSL_HOME%₩bin₩openssl.cnf
Signature ok
subject=C = KR, ST = Daejeon, L = Daejeon, O = DDIT, OU = CLASS_207, CN = lo
calhost
Getting CA Private Key
Enter pass phrase for rootca_private.key: java
```

인증서 = 인증서 소유자의 private key + 인증서 소유자의 public key + [인증서 발급자의 public key + 인증서 발급자의 private key] : 유효기간 5년 인증서 생성

-CA : 인증서를 발급할 CA 의 인증서 파일, 이 옵션을 사용하는 경우 CA 인증서의 subject name 이 최종 생성된 인증서의 issuer 로 설정된다.

-CAkey : 인증서에 서명할 CA 의 개인키

-CAcreateserial : CA 의 일련번호 파일 강제 생성

-extfile : extensions 정보를 가진 설정 파일(v3_req 섹션을 가진 openssl.cnf)

4) CA 가 발급한 인증서와 톰캣의 개인키로 keystore 생성
(https://www.openssl.org/docs/manmaster/man1/pkcs12.html)

① 톰캣의 keystore 생성 : JKS 나 PKCS12 표준을 이용한 keystore 생성
(https://tomcat.apache.org/tomcat-8.5-doc/ssl-howto.html
-> Prepare the Certificate Keystore)
PKCS12 : openSSL 과 MS 에서 사용되는 keystore 방식의 인증서 표준, CA 인증서와 서버의 인증서를 기반으로 keystore 를 생함.

① 톰캣의 keystore 생성 : JKS 나 PKCS12 표준을 이용한 keystore 생성(계속)

```
D:₩ncs_module₩cert>openssl pkcs12 -export -in localhost.crt -inkey localhost
_private.key -out keystore -name "tomcat cert" -CAfile rootca.crt -caname ro
ot -chain
Enter pass phrase for localhost_private.key:
Enter Export Password:                                    java
Verifying - Enter Export Password:
```

-export : keysotre 파일 생성

-out : keystore 파일명

-in : keysotre 파일 생성에 사용될 서버 인증서

-inkey : keystore 파일 생성에 사용될 서버의 개인키

-CAfile : 서버 인증서를 발급한 CA 인증서

-chain : 인증서의 전체 인증 구조(루트 CA 와 중간 CA 모두)를 포함하여 keystore 생성

② 로컬 서버에 https 설정 (%CATALINA_HOME%₩conf₩server.xml)

```
<Connector port="80" protocol="HTTP/1.1"
       connectionTimeout="20000"  redirectPort="443"
       useBodyEncodingForURI="true" />

<Connector port="443" protocol="org.apache.coyote.http11.Http11Protocol"
   maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
   keystoreFile="D:/ncs_module/cert/keystore" keystorePass="java"
   keystoreType="pkcs12" clientAuth="false" sslProtocol="TLS" />
```

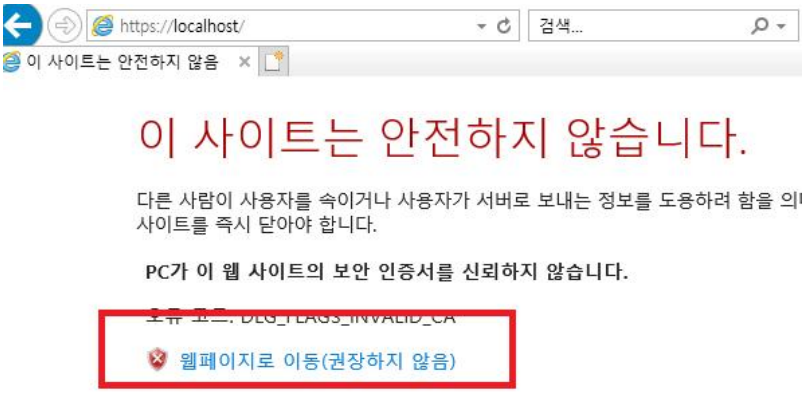
③ http 로 접속시 https 로 리다이렉션 설정(%CATALINA_HOME%₩web.xml)

```
<security-constraint>
        <web-resource-collection>
                <web-resource-name>SSL Forward</web-resource-name>
                <url-pattern>/*</url-pattern>
        </web-resource-collection>
        <user-data-constraint>
                <transport-guarantee>CONFIDENTIAL</transport-guarantee>
        </user-data-constraint>
</security-constraint>
```

최종적으로 생성된 로컬 서버용 인증서 확인

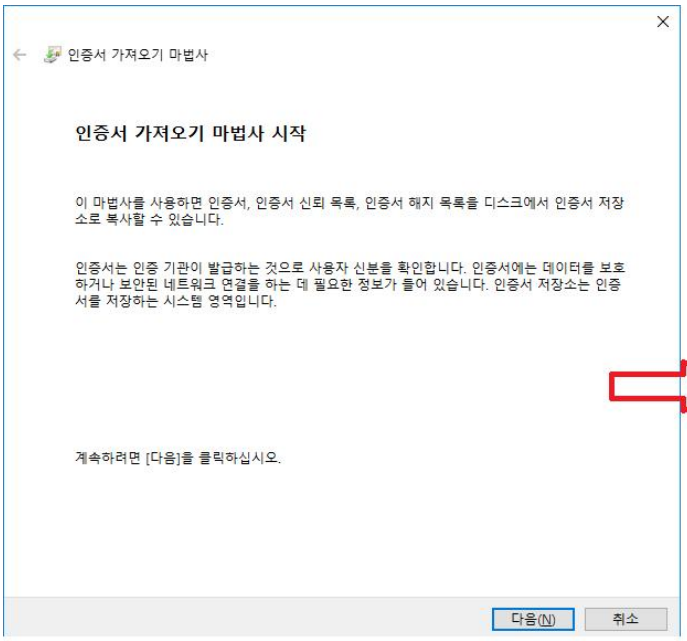
```
D:₩ncs_module₩cert>openssl x509 -text -in localhost.crt
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            e9:bd:36:99:55:7c:63:3f          인증서 발급자 == CA
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = KR, ST = Daejeon, L = Daejeon, O = DDIT, OU = DDIT_SUB,
CN = localhost
        Validity
            Not Before: May 14 07:23:14 2019 GMT
            Not After : May 12 07:23:14 2024 GMT   인증서 소유자 == local server
        Subject: C = KR, ST = Daejeon, L = Daejeon, O = DDIT, OU = CLASS_207
, CN = localhost
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:b5:b5:7d:94:d1:f5:31:f3:25:04:e1:77:45:24:
                    5f:38:32:b9:3a:b9:3a:cf:29:05:e7:7d:e8:79:43:
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 Subject Alternative Name:
                DNS:localhost, IP Address:127.0.0.1, IP Address:192.168.207.
06, IP Address:192.168.0.61          인증서 소유자의 domain/ip 정보
    Signature Algorithm: sha256WithRSAEncryption
                    4c:cb:56:4a:60:3f:e6:fc:1e:8f:84:09:7f:bd:96:
```

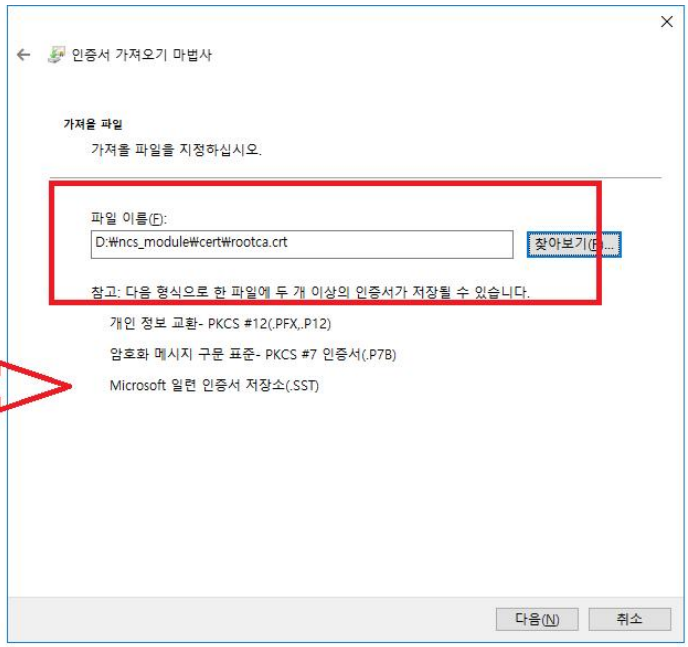
5) 테스트 클라이언트의 신뢰할수 있는 CA 리스트에 Root CA 를 직접 등록

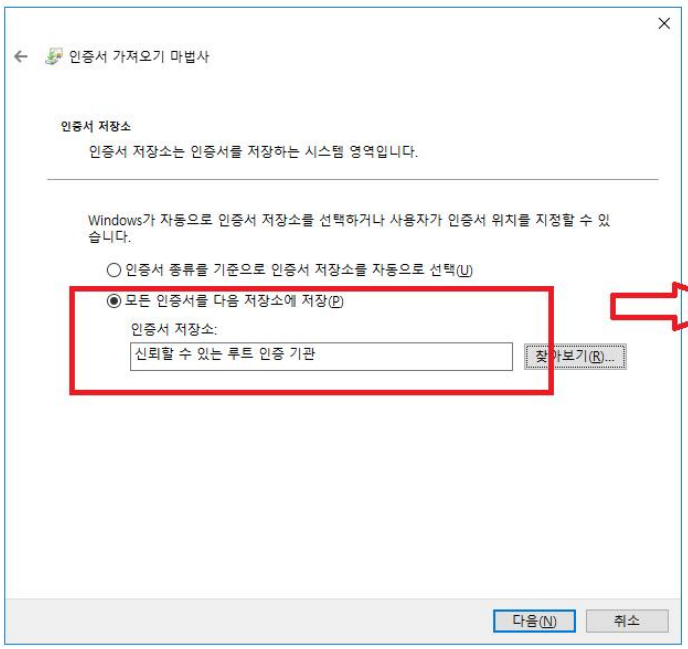


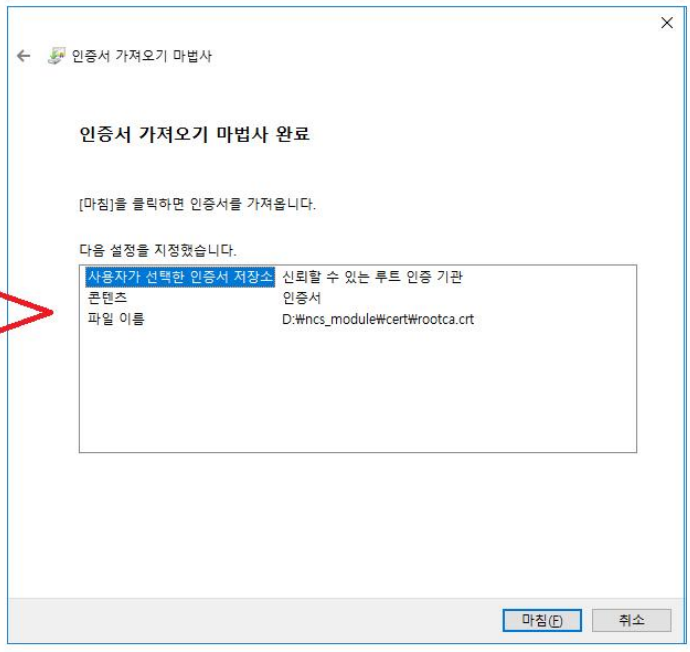
서버의 인증서를 발급한 Root CA 가 신뢰할 수 있는 CA 리스트에 포함되지 않기 때문에 서버의 인증서 검증 단계에서 오류가 발생한다.

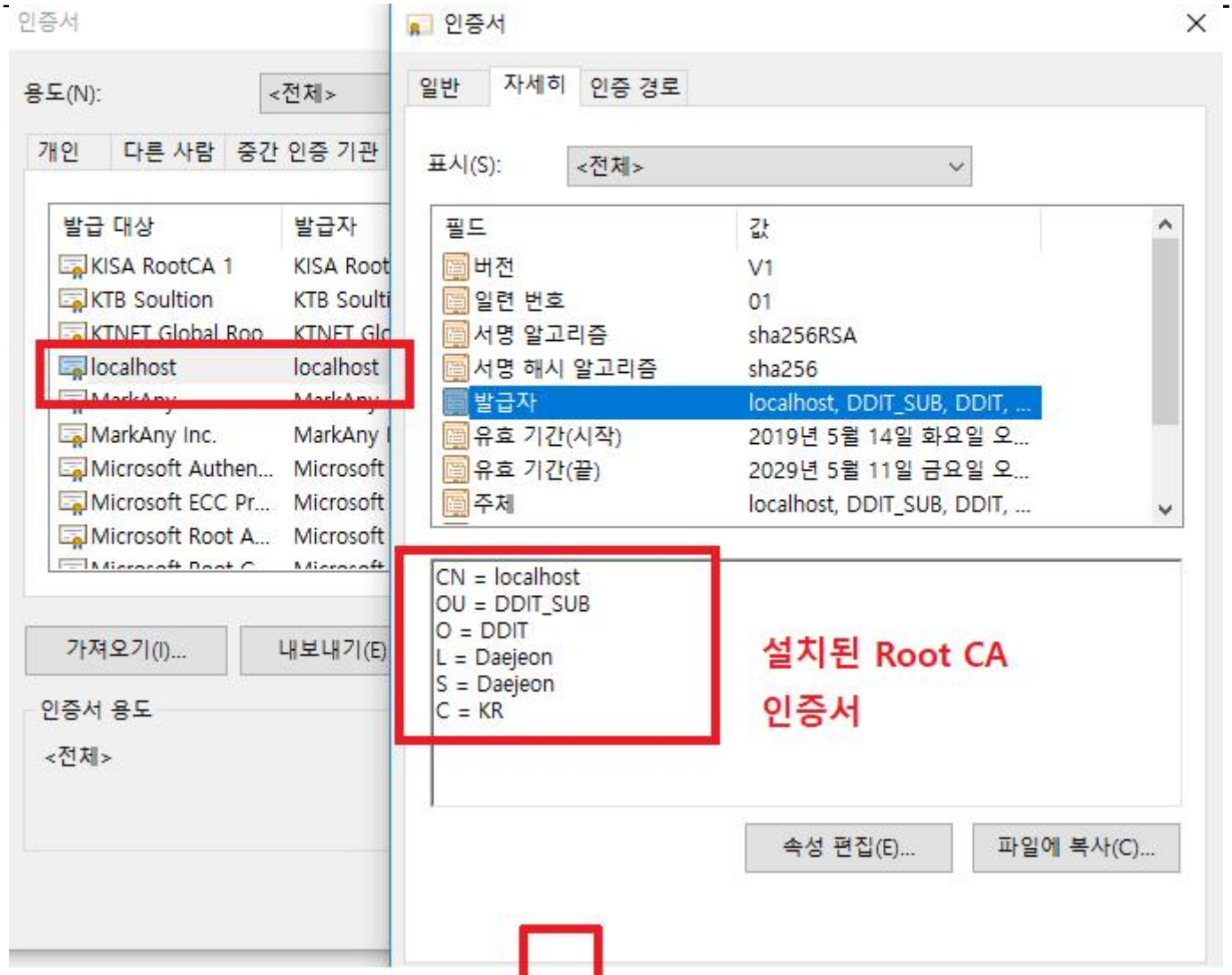
브라우저 메뉴바 -> 도구 -> 인터넷 옵션 -> 내용 -> 인증서 -> 신뢰할 수 있는 루트 인증 기관 -> 가져오기 -> rootca.crt 파일을 이용한 인증서 등록

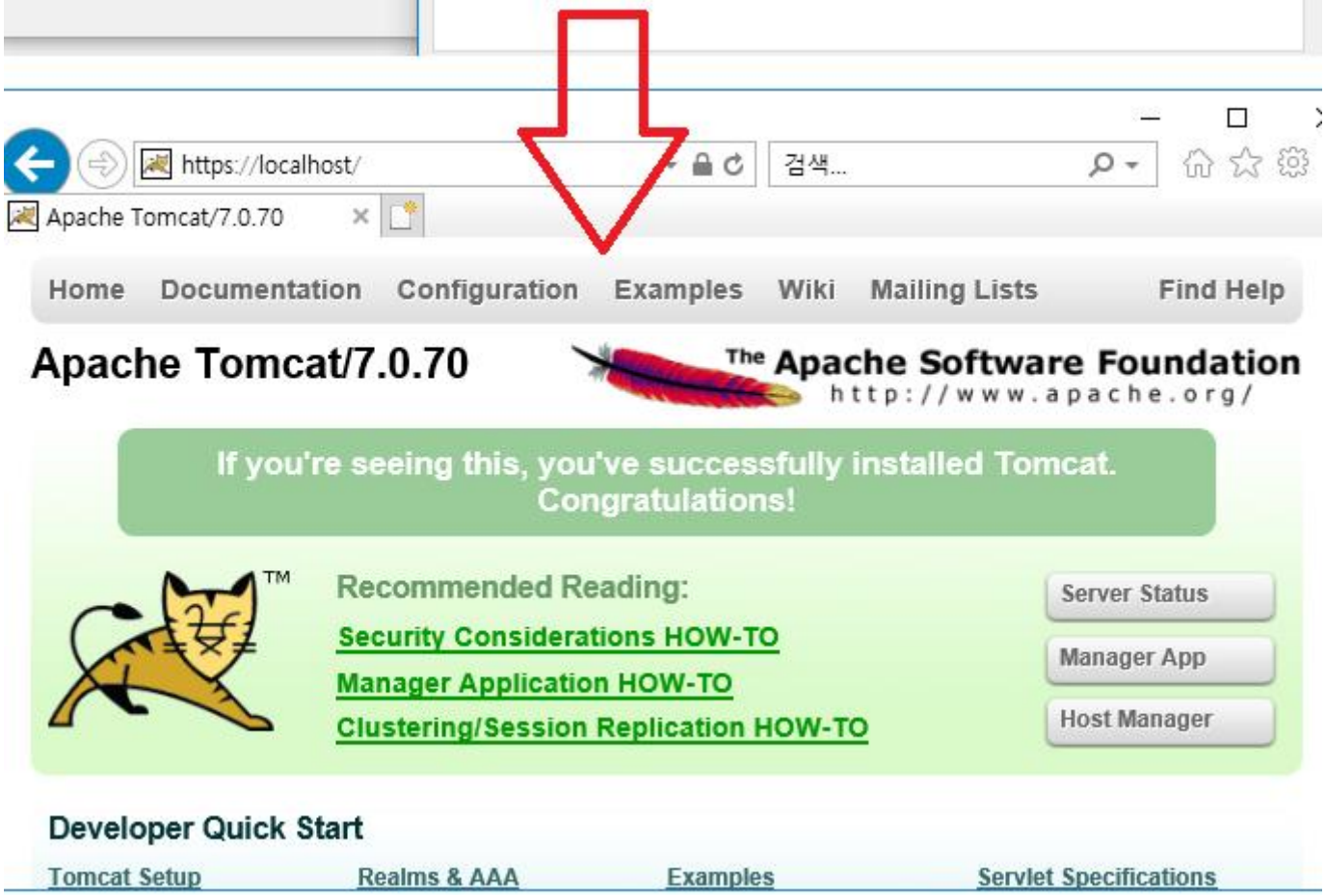









인증서
용도(N):
<전체>
개인
다른 사람
중간 인증 기관
발급 대상
발급자
KISA RootCA 1
KISA Root
KTB Soultion
KTB Soulti
localhost
localhost
MarkAny Inc.
Microsoft Authen...
Microsoft ECC Pr...
Microsoft Root A...
가져오기(I)...
내보내기(E)
인증서 용도
<전체>
인증서
일반
자세히
인증 경로
표시(S):
<전체>
필드
값
버전
V1
일련 번호
01
서명 알고리즘
sha256RSA
서명 해시 알고리즘
sha256
발급자
localhost, DDIT_SUB, DDIT, ...
유효 기간(시작)
2019년 5월 14일 화요일 오...
유효 기간(끝)
2029년 5월 11일 금요일 오...
주체
localhost, DDIT_SUB, DDIT, ...
CN = localhost
OU = DDIT_SUB
O = DDIT
L = Daejeon
S = Daejeon
C = KR
설치된 Root CA
인증서
속성 편집(E)...
파일에 복사(C)...


https://localhost/
검색...
Apache Tomcat/7.0.70
Home
Documentation
Configuration
Examples
Wiki
Mailing Lists
Find Help
Apache Tomcat/7.0.70
The Apache Software Foundation
http://www.apache.org/
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
Server Status
Manager App
Host Manager
Developer Quick Start
Tomcat Setup
Realms & AAA
Examples
Servlet Specifications

❑ 프로젝트 시연을 위한 SSL 설정

1. ₩₩sem-pc₩최종프로젝트참고₩rootCA 폴더 를 로컬에 복사
   ex) D:₩A_TeachingMaterial₩7.LastProject₩rootCA
   공인된 CA 를 사용할 수 없는 강의실 내 모든 로컬 서버들을 대상으로 인증서 발급에 공통 사용할 사설 CA 인증서 파일들을 배포함.
2. 배포된 openssl.pdf 파일을 참고하여 openssl toolkit 설치 및 환경변수 설정
   다운로드 -> 설치 -> 환경변수 설정(OPENSSL_HOME, OPENSSL_CONF, PATH)
   주의!
   1) openssl.cnf 파일의 v3_req 섹션에서 alt_names 에 사용될 가상 도메인은 c:/windows/system32/drivers/etc/hosts 파일에 팀 프로젝트 어플리케이션 도메인으로 미리 설정.
   2) 설치 도중 dll 파일을 %OPENSSL_HOME%₩bin 에 복사하는 옵션을 선택한 경우, " msvcr1XX.dll 파일 검색 관련 에러가 발생할 수 있으며, 이 경우 openssl.pdf 파일의 안내에 따라 visual c++ 패키지를 설치한다. (https://www.microsoft.com/ko-KR/download/details.aspx?id=40784)
3. openssl.pdf 파일의 인증서 생성 절차 5. 번부터 수행
   1) D:₩A_TeachingMaterial₩7.LastProject₩local_cert 폴더 생성
   2) 윈도우 콘솔을 관리자 권한으로 실행하고, 1) 번의 폴더로 이동한다.
   3) openssl.pdf -> 5. 자가 서명된 CA 로 발급할 로컬 서버(톰캣)의 인증서 생성 -> 1) 번부터 수행
      주의!
      ① rootCA 의 위치가 현재 폴더 기준 "../rootCA/rootca.crt" 이므로
      5.->3) 번의 localhost.crt 파일 생성시
      -CA rootca.crt -> -CA ../rootCA/rootca.crt
      -CAkey rootca_private.key -> –Cakey ../rootCA/rootca_private.key 로 변경됨

```
openssl x509 -req -days 1825 -in localhost.csr -out localhost.crt
-CA ../rootCA/rootca.crt -CAcreateserial -CAkey ../rootCA/.rootca_priv
ate.key -extensions v3_req -extfile %OPENSSL_HOME%₩bin₩openssl.cnf
```

      ② 5.->4) 번의 keystore 파일 생성 역시
      -CAfile rootca.crt -> -CAfile ../rootCA/rootca.crt 로 변경됨.

```
openssl pkcs12 -export -in localhost.crt -inkey localhost_private.
key -out keystore -name "tomcat cert" -CAfile ../rootCA/rootca.crt -ca
name root -chain
```

      ③ 5.->5) 단계에서 현재 폴더 기준 "../rootCA/rootca.crt" , 즉 공용으로 배포된 사설 CA 인증서로 신뢰할 수 있는 인증기관 정보를 가져옴.