

# NGINX SSL 프로토콜 가이드

본 문서는 주식회사 한국기업보안에서 SSL보안서버인증서 설치를 위해 작성된 문서로  
주식회사 한국기업보안의 동의 없이 무단으로 사용하실 수 없습니다.

[고객센터]

한국기업보안. 유서트 기술팀

02-3442-7230

UCERT



www.한국기업보안.kr  
Korea Corporation Security

#### [사용이 제외된 알고리즘]

TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_RC4_128_MD5
TLS_ECDHE_RSA_WITH_RC4_128_SHA

#### [취약한 알고리즘]

TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_RSA_WITH_AES_128_GCM_SHA256
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_CAMELLIA_256_CBC_SHA
TLS_RSA_WITH_CAMELLIA_128_CBC_SHA
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_DES_CBC_SHA
TLS_DHE_RSA_WITH_SEED_CBC_SHA

#### [사용을 권장하는 알고리즘]

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

#### [DHE 알고리즘 사용 권고]

DHE 알고리즘은 웹서버 환경인 OPENSSL 및 Java 버전의 따라 취약한 비트수를 사용하게 됩니다. 비트 수는 1024, 2048bit를 사용하며 1024bit의 경우 약한 알고리즘이며 **2048bit의 경우가 안전한 알고리즘을 사용** 합니다. 아래의 표는 2048bit를 출력하는 OPENSSL 버전과 Java 버전 입니다.

OPENSSL	Java
OpenSSL 1.0.2 : openssl 1.0.2b 이상 OpenSSL 1.0.1 : openssl 1.0.1n 이상	JDK 1.8 이상



## 설정 확인 사항.

### 1. 권장 알고리즘과 범용 알고리즘 중 어느 알고리즘을 적용해야 하나요?

권장 알고리즘은 높은 수준의 Cipher Suites(SSL 알고리즘)입니다. 높은 등급의 보안 점수를 받는데 도움이 됩니다. 문제는 낮은 수준의 보안 설정만 지원되는 사용자 단말기에서는 접속이 되지 않을 수 있습니다. 또한 웹 서버에서도 높은 수준의 보안 사양을 가지고 있어야 합니다.

범용 알고리즘은 취약한 알고리즘을 포함한 범용성을 위한 알고리즘입니다. 높은 수준의 알고리즘을 사용 못하는 클라이언트가 있을 경우 사용하기 위한 알고리즘입니다. 낮은 수준의 보안 등급과 함께 취약성이 드러나 있으므로 주의 바랍니다.

### 2. 적용하기 전 주의사항이 있을까요?

각 웹 서버에 대한 재 시작이 필요합니다. 다만, 적용하시기 전 알고리즘이 지원되는지 확인이 필요합니다.

IIS를 사용하는 윈도우 서버는 레지스트리를 수정해야 함으로 각별한 주의가 필요합니다.

### 3. 적용을 하고 나면 앞으로 같은 작업을 할 필요가 없나요?

그렇지 않습니다. 알고리즘의 취약점, 프로토콜에 관한 취약점이 발견되면 이와 같은 사항을 다시 안내 드릴 수 있습니다.

### 4. 프로토콜과 알고리즘은 별개로 조정을 해도 괜찮은가요?

가능합니다. 다만, 프로토콜과 알고리즘은 매우 밀접한 관계로 프로토콜 지원이 가능하더라도 TLS 1.2에 지원가능 알고리즘이 없으면 TLS 1.2 통신이 되지 않습니다. 반대로 높은 수준의 알고리즘은 TLS 1.2에서만 지원됩니다(ECDHE, GCM, SHA256, SHA384 등등).

## 1. NGINX SSL Protocol 및 Cipher 설정

### - 권장 알고리즘

- (1) Docker 안의 nginx 설정 문서 (\$HOME\_BASE)/**nginx.conf** 파일을 열어 프로토콜을 아래와 같이 설정 합니다. 설정문서는 환경에 따라 다른 위치, 파일일 수 있습니다.

```
server {  
    listen      443 ssl;  
    server_name localhost;  
  
    ssl_certificate      /etc/nginx/ucert/sslinstall.ucert.co.kr.pem;  
    ssl_certificate_key  /etc/nginx/ucert/sslinstall.ucert.co.kr.key;  
  
    ssl_session_cache    shared:SSL:1m;  
    ssl_session_timeout  5m;  
  
    ssl_protocols TLSv1.2;  
    ssl_ciphers HIGH:!aNULL:!MD5;  
    ssl_prefer_server_ciphers on;  
  
    location / {  
        root    html;  
        index  index.html index.htm;  
    }  
}
```

- (2) (\$HOME\_BASE)/**nginx.conf** 파일을 열어 알고리즘을 아래와 같이 설정 합니다.  
(**ssl\_ciphers** 값은 zip 파일 안에 동봉된 텍스트 파일(txt)에서 복사하셔서 사용하십시오.)

#### **ssl\_ciphers**

```
EECDH+ECDSA+AESGCM:EECDH+aRSA+AESGCM:EECDH+ECDSA+SHA256:EECDH+aRSA+SHA256:E  
ECDH+ECDSA+SHA384:EECDH+ECDSA+SHA256:EECDH+aRSA+SHA384:EDH+aRSA+AESGCM:EDH+  
aRSA+SHA256:EDH+aRSA:EECDH:!aNULL:!eNULL:!MEDIUM:!LOW:!3DES:!MD5:!EXP:!PSK:!SRP:!DSS:!  
RC4:!SEED;
```

『적용 알고리즘』

- (3) 위의 구문들을 추가하여 다음과 같이 설정 합니다.

```
server {  
    listen      443 ssl;
```



```
server_name localhost;
```

```
ssl_certificate /etc/nginx/ssl/sslinstall.ucert.co.kr.pem;
```

```
ssl_certificate_key /etc/nginx/ssl/sslinstall.ucert.co.kr.key;
```

```
ssl_session_cache shared:SSL:1m;
```

```
ssl_session_timeout 5m;
```

```
ssl_protocols TLSv1.2;
```

```
ssl_ciphers EECDH+ECDSA+AESGCM:EECDH+aRSA+AESGCM:EECDH+ECDSA+SHA256:  
EECDH+aRSA+SHA256:EECDH+ECDSA+SHA384:EECDH+ECDSA+SHA256:EECDH+aRSA+SHA  
384:EDH+aRSA+AESGCM:EDH+aRSA+SHA256:EDH+aRSA:EECDH:!aNULL:!eNULL:!MEDIUM:!L  
OW:!3DES:!MD5:!EXP:!PSK:!SRP:!DSS:!RC4:!SEED;
```

```
ssl_prefer_server_ciphers on;
```

```
location / {
```

```
    root html;
```

```
    index index.html index.htm;
```

```
}
```

```
}
```

『설정 예시』

## 2. NGINX SSL Protocol 및 Cipher 설정

### - 범용 알고리즘

(1) (\$HOME\_BASE)/**nginx.conf** 파일을 열어 프로토콜을 아래와 같이 설정 합니다.

```
server {
```

```
    listen 443 ssl;
```

```
    server_name localhost;
```

```
    ssl_certificate /etc/nginx/ucert/sslinstall.ucert.co.kr.pem;
```

```
    ssl_certificate_key /etc/nginx/ucert/sslinstall.ucert.co.kr.key;
```

```
    ssl_session_cache shared:SSL:1m;
```

```
    ssl_session_timeout 5m;
```

```
ssl_protocols TLSv1.2;
```

```
ssl_ciphers HIGH:!aNULL:!MD5;
```

```
ssl_prefer_server_ciphers on;
```



```

location / {
    root    html;
    index  index.html index.htm;
}

```

- (3) (\$HOME\_BASE)/**nginx.conf** 를 열어 알고리즘을 아래와 같이 설정 합니다.  
**(ssl\_ciphers** 값은 zip 파일 안에 동봉된 텍스트 파일(txt)에서 복사하셔서 사용하십시오.)

```

ssl_ciphers    ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-
DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-
SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-
ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-DSS-AES128-
SHA256:DHE-DSS-AES256-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA:AES256-
SHA:AES:CAMELLIA:!DES-CBC3-SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-
DES-CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-DES-CBC3-SHA:!DHE-RSA-AES128-GCM-
SHA256:!DHE-RSA-AES256-GCM-SHA384:!DHE-RSA-AES128-SHA256:!DHE-RSA-AES256-SHA:!DHE-
RSA-AES128-SHA:!DHE-RSA-AES256-SHA256:!DHE-RSA-CAMELLIA128-SHA:!DHE-RSA-CAMELLIA256-
SHA:!DHE-RSA-3DES;

```

『적용 알고리즘』

- (3) 위의 구문들을 추가하여 다음과 같이 설정 합니다.

```

server {
    listen      443 ssl;
    server_name localhost;

    ssl_certificate      /etc/nginx/ucert/sslinstall.ucert.co.kr.pem;
    ssl_certificate_key  /etc/nginx/ucert/sslinstall.ucert.co.kr.key;

    ssl_session_cache    shared:SSL:1m;
    ssl_session_timeout  5m;

    ssl_protocols TLSv1.2;
    ssl_ciphers    ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-
GCM-SHA384:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-
SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-

```



```

SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-
SHA:ECDHE-ECDSA-AES256-SHA:DHE-DSS-AES128-SHA256:DHE-DSS-AES256-SHA:AES128-
GCM-SHA256:AES256-GCM-SHA384:AES128-SHA:AES256-SHA:AES:CAMELLIA:!DES-CBC3-
SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-DES-CBC3-SHA:!EDH-
RSA-DES-CBC3-SHA:!KRB5-DES-CBC3-SHA:!DHE-RSA-AES128-GCM-SHA256:!DHE-RSA-
AES256-GCM-SHA384:!DHE-RSA-AES128-SHA256:!DHE-RSA-AES256-SHA:!DHE-RSA-AES128-
SHA:!DHE-RSA-AES256-SHA256:!DHE-RSA-CAMELLIA128-SHA:!DHE-RSA-CAMELLIA256-
SHA:!DHE-RSA-3DES;
ssl_prefer_server_ciphers on;

location / {
    root    html;
    index   index.html index.htm;
}
}

```

『설정 예시』

(4) 작업을 진행하기 전 반드시 기존 설정 파일을 백업 진행토록 합니다.

```

[root@localhost nginx]# cp nginx.conf nginx.conf.backup

[root@localhost nginx]# ll nginx.conf*

total 2

-rw-r--r--. 1 root root 2545 May 17 08:16 nginx.conf

-rw-r--r--. 1 root root 2545 May 27 08:22 nginx.conf.backup

```

『복원 예시』

### 3. Docker – NGINX 재기동

(1) Docker의 nginx container를 재기동 하여 줍니다.

기본적으로 docker stop/start <컨테이너> 혹은 restart <컨테이너> 로 진행됩니다.

```

[root@localhost ~]# docker ps

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
NAMES					



```
550ae502ce01    nginx    "/docker-entrypoint...."    5 days ago    Up 26 hours    0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp, 4443/tcp    nginx
```

```
[root@localhost ~]# docker stop nginx1
```

```
[root@localhost ~]# docker start nginx1
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
550ae502ce01	nginx	"/docker-entrypoint...."	5 days ago	Up 5 seconds	0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp, 4443/tcp

## 4. TLS 적용 확인 방법

(1) OPENSSSL 을 이용한 확인 방법 입니다.

- OpenSSL은 네트워크를 통한 데이터 통신에 쓰이는 프로토콜인 TLS와 SSL의 오픈 소스 입니다. 기본적인 암호화 기능 및 여러 유틸리티 함수들이 구현되어 있습니다.

openssl s_client -connect [해당 IP 및 도메인]:443 -ssl3	SSLv3 프로토콜 통신 확인.
openssl s_client -connect [해당 IP 및 도메인]:443 -ssl2	SSLv2 프로토콜 통신 확인.
openssl s_client -connect [해당 IP 및 도메인]:443 -tls1	TLSv1.0 프로토콜 통신 확인.
openssl s_client -connect [해당 IP 및 도메인]:443 -tls1_1	TLSv1.1 프로토콜 통신 확인.
openssl s_client -connect [해당 IP 및 도메인]:443 -tls1_2	TLSv1.2 프로토콜 통신 확인.

『적용 명령어』

- 아래의 명령어를 이용하여 SSLv3에 대한 통신이 제한 되었는지 확인 합니다.

```
openssl s_client -connect 127.0.0.1:443 -ssl3
```

『적용 예시』

```
CONNECTED(00000003)
140430722205512:error:14094410:SSL routines:SSL3_READ_BYTES:ssl3 alert handshake
failure:s3_pkt.c:1275:SSL alert number 40
```





```
140430722205512:error:1409E0E5:SSL routines:SSL3_WRITE_BYTES:ssl handshake failure:s3_pkt.c:598:
```

```
---
```

```
no peer certificate available
```

```
---
```

```
No client certificate CA names sent
```

```
---
```

```
SSL handshake has read 7 bytes and written 0 bytes
```

```
---
```

```
New, (NONE), Cipher is (NONE)
```

```
Secure Renegotiation IS NOT supported
```

```
Compression: NONE
```

```
Expansion: NONE
```

```
SSL-Session:
```

```
Protocol : SSLv3
```

```
Cipher : 0000
```

```
Session-ID:
```

```
Session-ID-ctx:
```

```
Master-Key:
```

```
Key-Arg : None
```

```
Krb5 Principal: None
```

```
PSK identity: None
```

```
PSK identity hint: None
```

```
Start Time: 1558603004
```

```
Timeout : 7200 (sec)
```

```
Verify return code: 0 (ok)
```

『적용 결과』

- 위 명령의 결과에서 **Renegotiation IS NOT supported** 구문을 확인하여 해당 프로토콜의 사용이 제한된 것을 확인할 수 있습니다.

- 아래의 명령어를 이용하여 TLSv1.2에 대한 통신이 제한 되었는지 확인 합니다.

```
openssl s_client -connect localhost:443 -tls1_2
```

『적용 예시』

```
New, TLSv1/SSLv3, Cipher is AES128-GCM-SHA256
```

```
Server public key is 2048 bit
```

```
Secure Renegotiation IS supported
```

```
Compression: NONE
```



Expansion: NONE

SSL-Session:

Protocol : TLSv1.2

Cipher : AES128-GCM-SHA256

Session-ID: CA6B7CB11138789ABE9B7FEB37EE08A3F57E58E1952B0A545B8AE6D78218F9E1

Session-ID-ctx:

Master-Key:

F2E89E039E0617B88B0D7176E7A8052DDD6064564D1808126456D6825186C28DB30BA1824D9898E4E2B  
A4C38C7D7A6A6

Key-Arg : None

Krb5 Principal: None

PSK identity: None

PSK identity hint: None

TLS session ticket lifetime hint: 300 (seconds)

TLS session ticket:

0000 - 7f f1 06 59 83 a5 b0 4d-42 1f f8 26 66 b6 04 e6 ...Y...MB..&f...  
0010 - 2d 9d 68 51 55 98 ed 9e-bf 9c 4c 41 f1 a3 3f 1a -.hQU.....LA..?.  
0020 - e7 24 a9 f6 84 f5 4a 7e-61 4f 25 7a 4a b6 4e cc \$.....J~aO%zJ.N.  
0030 - ed dc 29 6d 52 03 17 92-49 32 17 d9 27 10 76 9b ..)mR...l2..'v.  
0040 - 68 27 16 8a c7 ad 20 09-f0 c1 03 03 92 15 da 68 h'.... .....h  
0050 - d9 cf e7 14 f8 f9 aa 0d-61 30 63 46 49 0d 50 bc .....a0cFI.P.  
0060 - cb 93 ab b5 9c 9f a4 41-02 c4 0f 04 2c 58 93 43 .....A....,X.C  
0070 - a3 8e 08 d1 20 86 4b 8e-ca 39 51 95 9d b1 bc b8 .... .K..9Q....  
0080 - 2b 80 11 04 fc 7f 7a de-77 b2 2e 0d 97 37 18 55 +.....z.w....7.U  
0090 - e6 09 e8 3a 0e f5 49 94-fb c4 cf 9a bb c9 51 a9 ....l.....Q  
00a0 - c9 be ec c8 e9 77 f2 12-bf 3c 8f 70 0d 65 73 e2 .....w...<.p.es.  
00b0 - d9 5a 03 bb ec 48 18 76-76 75 48 44 0c d2 15 bc .Z...H.vvuHD....

Start Time: 1558603741

Timeout : 7200 (sec)

Verify return code: 0 (ok)

『적용 결과』

- 위 명령의 결과에서 **Renegotiation IS NOT supported** 구문을 확인하여 해당 프로토콜의 사용이 제한된 것을 확인할 수 있습니다.

(2) Nmap 을 이용한 확인 방법 입니다.

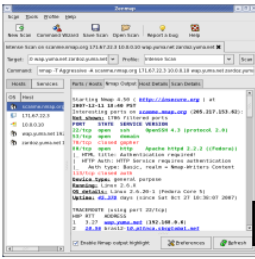
- Nmap 은 서비스 탐지 프로토콜로 자신을 광고하지 않는 수동적인 서비스들을 찾아낼 수 있습니다. 이 상세 정보에는 운영 체제, 장치 종류, 운영 시간, 서비스에 쓰이는 소프트웨어 및 제품의 정확한 버전, 방화벽 기술의 존재와 심지어 근거리 네트워크에서 네트워크 카드의 공급자도 포함 됩니다.



- 간단한 UI 환경을 가지고 있으며 아래와 같은 방법으로 진행을 합니다. 아래의 URL에서 프로그램을 다운로드 합니다.

<https://nmap.org/download.html>

Microsoft Windows binaries



Please read the [Windows section](#) of the Install Guide for limitations and installation instructions for the Windows Zenmap GUI or the much smaller command-line zip file version. We support Nmap on Windows 7 and newer, [Nmap on earlier Windows releases](#).

**Note:** The version of Npcap included in our installers may not always be the latest version. If you experience problems, please see the [Npcap release page](#).

The Nmap **executable Windows installer** can handle Npcap installation, registry performance tweaks, and detect Zenmap graphical frontend. Skip all the complexity of the Windows zip files with a self-installer:

**Latest stable release self-installer: [nmap-7.70-setup.exe](#)**

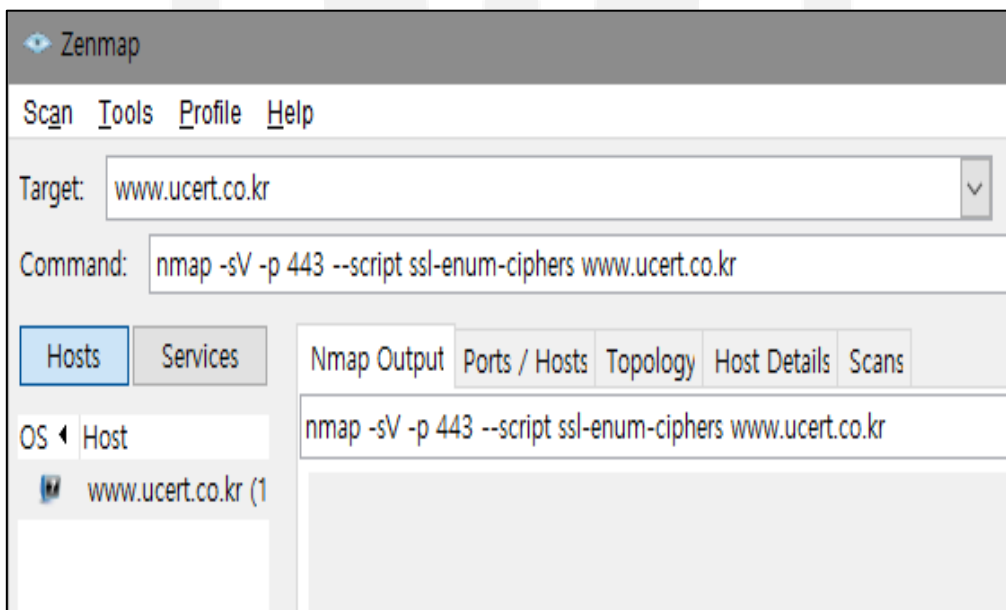
**Latest Npcap release self-installer: [npcap-0.99-r7.exe](#)**

We have written [post-install usage instructions](#). Please [notify us](#) if you encounter any problems or have suggestions for the installer.

For those who prefer the command-line zip files ([Installation Instructions](#); [Usage Instructions](#)), they are still available. The Zenmap graphical interface is available as a separate download. Or you can download and install a superior command shell such as those included with the free [Cygwin system](#). Also, you need to have a C++ compiler installed in the zip file. The main advantage is that these zip files are a fraction of the size of the executable installer:

**Latest stable command-line zipfile: [nmap-7.70-win32.zip](#)**

『설치 프로그램 다운로드』



- Nmap을 실행한 후에 'Command' 창에 명령어를 입력 합니다.

```
nmap -sV --script ssl-enum-ciphers -p <포트번호> <host>
```

『적용 명령어』

- 포트번호에 확인 할 포트를 기입 합니다. Target에는 도메인을 기입 합니다.

```
nmap -sV --script ssl-enum-ciphers -p 443 121.78.88.70
```

『적용 예시』



본 문서는 주식회사 한국기업보안에서 SSL보안서버인증서 설치를 위해 작성된 문서로  
주식회사 한국기업보안의 동의 없이 무단으로 사용하실 수 없습니다

Copyright 2018-2023. Korea Corporation Security Co., Ltd All pictures cannot be copied without permission.

PORT     STATE SERVICE VERSION

443/tcp open    ssl/ssl Apache httpd (SSL-only mode)

|\_http-server-header: Apache

| ssl-enum-ciphers:

|    TLSv1.0:

|     ciphers:

|       TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA (rsa 2048) - A

|       TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA (rsa 2048) - A

|       TLS\_RSA\_WITH\_CAMELLIA\_256\_CBC\_SHA (rsa 2048) - A

|       TLS\_RSA\_WITH\_CAMELLIA\_128\_CBC\_SHA (rsa 2048) - A

|       TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA (rsa 2048) - C

|     compressors:

|       NULL

|     cipher preference: server

|     warnings:

|       64-bit block cipher 3DES vulnerable to SWEET32 attack

|    TLSv1.1:

|     ciphers:

|       TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA (rsa 2048) - A

|       TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA (rsa 2048) - A

|       TLS\_RSA\_WITH\_CAMELLIA\_256\_CBC\_SHA (rsa 2048) - A

|       TLS\_RSA\_WITH\_CAMELLIA\_128\_CBC\_SHA (rsa 2048) - A

|       TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA (rsa 2048) - C

|     compressors:

|       NULL

|     cipher preference: server

|     warnings:

|       64-bit block cipher 3DES vulnerable to SWEET32 attack

|    TLSv1.2:

|     ciphers:

|       TLS\_RSA\_WITH\_AES\_128\_GCM\_SHA256 (rsa 2048) - A

|       TLS\_RSA\_WITH\_AES\_256\_GCM\_SHA384 (rsa 2048) - A

|       TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA (rsa 2048) - A

|       TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA (rsa 2048) - A

|       TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256 (rsa 2048) - A



```
| TLS_RSA_WITH_AES_128_CBC_SHA256 (rsa 2048) - A
| TLS_RSA_WITH_CAMELLIA_256_CBC_SHA (rsa 2048) - A
| TLS_RSA_WITH_CAMELLIA_128_CBC_SHA (rsa 2048) - A
| TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 2048) - C
| compressors:
|   NULL
| cipher preference: server
| warnings:
|   64-bit block cipher 3DES vulnerable to SWEET32 attack
```

『적용 결과』

- 적용 결과 알고리즘과 함께 사용중인 프로토콜을 확인 할 수 있습니다.

UCERT

[www.ucert.co.kr](http://www.ucert.co.kr)

