TLS 暗号設定 サーバ設定編

Ver 1.1

令和 3 年 12 月

独立行政法人 情報処理推進機構

目次

本書では、サーバ設定を行う上での参考情報として、設定方法例を記載する。

なお、利用するバージョンやディストリビューションの違いにより、設定方法が異なったり、設定ができなかったりする場合があることに留意すること。正式な取扱説明書やマニュアルを参照するとともに、一参考資料として利用されたい。

本書は以下のソフトウェアバージョンを対象として作成された。

OpenSSL 1.1.1d
GnuTLS 3.6.11.1
Apache httpd 2.4.41
lighttpd 1.4.54
nginx 1.16.1, 1.17.6
mod_gnutls 0.9.1

1. サーバ設定方法例のまとめ

1.1. Apache+mod_ssl の場合

Apache HTTP Server の設定ファイル（デフォルトの場合、httpd-ssl.conf）での設定例を以下に示す。

```
<VirtualHost *:443>
        （中略）
      SSLEngine on

    # 証明書と鍵の設定 [1]
    SSLCertificateFile /etc/ssl/chain.crt
    SSLCertificateKeyFile /etc/ssl/server.key

    # 暗号スイート設定。暗号スイートの設定例 1 章も参照のこと
    SSLCipherSuite "TLS1.2 以前用暗号スイート設定文字列"
    SSLCipherSuite TLSv1.3 "TLS1.3 用暗号スイート設定文字列"

    # プロトコルバージョン設定。2.1 節も参照のこと
    SSLProtocol バージョン設定

    # 暗号スイート順序サーバ優先設定
    SSLHonorCipherOrder On
```

[1] 設定する内容は以下のとおり。
/etc/ssl/chain.crt：サーバ証明書および中間証明書
/etc/ssl/server.key：サーバ証明書に対応する秘密鍵

HTTP Strict Transport Security、OCSP stapling の設定をする場合には、ここに追記する。ガイドライン 7.4 節及び 本書 5 章以降も参照のこと

</VirtualHost>

1.2. lighttpd の場合

lighttpd の設定ファイル（デフォルトの場合 lighttpd.conf ）での設定例を以下に示す。

```
$SERVER["socket"] == "0.0.0.0:443" {
    ssl.engine = "enable"
    （中略）

    # 証明書と鍵の設定
    ssl.pemfile = "/etc/ssl/serverkey_cert.pem"

    # 暗号スイート設定。暗号スイートの設定例 1 章も参照のこと
    ssl.cipher-list = "TLS1.2 以前用暗号スイート設定文字列"
    ssl.openssl.ssl-conf-cmd = ("Ciphersuites" => "TLS1.3 用暗号スイート設定文字列")[2]

    # プロトコルバージョン設定。2.2 節も参照のこと
    ssl.openssl.ssl-conf-cmd = ("MinProtocol" => バージョン設定, "MaxProtocol" => バージョン設定)[2]

    # 暗号スイート順序サーバ優先設定
    ssl.honor-cipher-order = "enable"

    HTTP Strict Transport Security の設定をする場合には、ここに追記する。ガイドライン 7.4 節及び本書 5 章以降を参照のこと。なお、lighttpd では OCSP stapling の設定はできない

}
```

[2] TLS1.3 暗号スイートとプロトコルバージョンの設定を共に行う場合は、一つの ssl.openssl.ssl-conf-cmd 内で双方の設定を行うこと

1.3. nginx の場合

nginx の設定ファイル（デフォルトの場合、nginx.conf）での設定例を以下に示す。

```
server {
    listen 443 ssl;
    （中略）

    # 証明書と鍵の設定
    ssl_certificate /etc/ssl/chain.crt;
    ssl_certificate_key /etc/ssl/server.key;

    # 暗号スイート設定。暗号スイートの設定例 1 章も参照のこと（特に TLS1.3 暗号スイートの設定については暗号スイートの設定例 1.1.2 節を参照のこと）
    ssl_ciphers "TLS1.2 以前用暗号スイート設定文字列";

    # プロトコルバージョン設定。2.3 節も参照のこと
    ssl_protocols プロトコルバージョン設定;

    # 暗号スイート順序サーバ優先設定
    ssl_prefer_server_ciphers on;


    HTTP Strict Transport Security、OCSP stapling の設定をする場合には、ここに追記する。ガイドライン 7.4 節及び本書 5 章以降を参照のこと

}
```

1.4. Apache+mod_gnutls の場合

Apache+mod_gnutls の設定ファイルでの設定例を以下に示す。

```
<VirtualHost *:443>
    （中略）
    GnuTLSEnable on

    # 証明書と鍵の設定
    GnuTLSCertificateFile /etc/ssl/chain.crt
    GnuTLSKeyFile /etc/ssl/server.key
```

暗号スイート、プロトコルバージョン、暗号スイート順序サーバ優先設定。2.4 節、3.4 節、暗号スイートの設定例 2 章も参照のこと

GnuTLSPriorities "暗号スイート設定文字列"

HTTP Strict Transport Security、OCSP stapling の設定をする場合には、ここに追記する。ガイドライン 7.4 節及び 本書 5 章以降も参照のこと

</VirtualHost>

2. プロトコルバージョンの設定方法例

2.1. Apache+mod_ssl の場合

プロトコルバージョンの設定は、1.1 節のプロトコルバージョン設定の部分に以下の内容を記述する。

- 推奨セキュリティ型
  SSLProtocol TLSv1.2 +TLSv1.3

- 推奨セキュリティ型（TLS1.3 無効）
  SSLProtocol TLSv1.2

- 高セキュリティ型
  SSLProtocol TLSv1.2 +TLSv1.3

- セキュリティ例外型
  SSLProtocol TLSv1 +TLSv1.1 +TLSv1.2 +TLSv1.3

- セキュリティ例外型（TLS1.0 無効）
  SSLProtocol TLSv1.1 +TLSv1.2 +TLSv1.3

- セキュリティ例外型（TLS1.0、TLS1.1 無効）
  SSLProtocol TLSv1.2 +TLSv1.3

- セキュリティ例外型（TLS1.3 無効）
  SSLProtocol TLSv1 +TLSv1.1 +TLSv1.2

2.2. lighttpd の場合

プロトコルバージョンの設定は、1.2 節のプロトコルバージョン設定の部分に以下の内容を記述する。

- 推奨セキュリティ型
  ssl.openssl.ssl-conf-cmd = ("MinProtocol" => "TLSv1.2", "MaxProtocol" => "TLSv1.3")

- 推奨セキュリティ型（TLS1.3 無効）
  ssl.openssl.ssl-conf-cmd = ("MinProtocol" => "TLSv1.2", "MaxProtocol" => "TLSv1.2")

- 高セキュリティ型
  ssl.openssl.ssl-conf-cmd = ("MinProtocol" => "TLSv1.2", "MaxProtocol" => "TLSv1.3")

- セキュリティ例外型
  ssl.openssl.ssl-conf-cmd = ("MinProtocol" => "TLSv1", "MaxProtocol" => "TLSv1.3")

- セキュリティ例外型（TLS1.0 無効）
  ssl.openssl.ssl-conf-cmd = ("MinProtocol" => "TLSv1.1", "MaxProtocol" => "TLSv1.3")

- セキュリティ例外型（TLS1.0、TLS1.1 無効）
  ssl.openssl.ssl-conf-cmd = ("MinProtocol" => "TLSv1.2", "MaxProtocol" => "TLSv1.3")

- セキュリティ例外型（TLS1.3 無効）
  ssl.openssl.ssl-conf-cmd = ("MinProtocol" => "TLSv1", "MaxProtocol" => "TLSv1.2")

2.3. nginx の場合

プロトコルバージョンの設定は、1.3 節のプロトコルバージョン設定の部分に以下の内容を記述する。

- 推奨セキュリティ型
  ssl_protocols TLSv1.2 TLSv1.3;

- 推奨セキュリティ型（TLS1.3 無効）
  ssl_protocols TLSv1.2;

- 高セキュリティ型
  ssl_protocols TLSv1.2 TLSv1.3;

- セキュリティ例外型
  ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;

- セキュリティ例外型（TLS1.0 無効）
  ssl_protocols TLSv1.1 TLSv1.2 TLSv1.3;

- セキュリティ例外型（TLS1.0、TLS1.1 無効）
  ssl_protocols TLSv1.2 TLSv1.3;

- セキュリティ例外型（TLS1.3 無効）
  ssl_protocols TLSv1 TLSv1.1 TLSv1.2;

2.4. Apache+mod_gnutls の場合

Apache+mod_gnutls では GnuTLSPriorities に指定する暗号スイート等の設定中にプロトコルバージョンに関する設定が含まれる。プロトコルバージョンを指定するためには、VERS-TLS で始まるプロトコルバージョンの設定に関する部分を変更する。

例えば、推奨セキュリティ型において TLS1.3 を無効にする場合は以下のように変更する。（太字部が変更箇所）

（変更前）
GnuTLSPriorities NONE:%SERVER_PRECEDENCE:**+VERS-TLS1.2:+VERS-TLS1.3**:+SIGN-ALL:+COMP-NULL:+GROUP-ALL:+ECDHE-ECDSA:+ECDHE-RSA:+DHE-RSA:+AES-128-GCM:+AES-256-GCM:+CHACHA20-POLY1305:+AES-128-CCM:+AES-256-CCM:+AES-128-CCM-8:+AES-256-CCM-8:+CAMELLIA-128-GCM:+CAMELLIA-256-GCM:+AEAD:+SHA384:+SHA256

（変更後）
GnuTLSPriorities NONE:%SERVER_PRECEDENCE:**+VERS-TLS1.2**:+SIGN-ALL:+COMP-NULL:+GROUP-ALL:+ECDHE-ECDSA:+ECDHE-RSA:+DHE-RSA:+AES-128-GCM:+AES-256-GCM:+CHACHA20-POLY1305:+AES-128-CCM:+AES-256-CCM:+AES-128-CCM-8:+AES-256-CCM-8:+CAMELLIA-128-GCM:+CAMELLIA-256-GCM:+AEAD:+SHA384:+SHA256

各型に対応するプロトコルバージョンの設定例を以下に示す。ただし、VERS-TLS で開始しない、プロトコルバージョン以外の設定要素は「(略)」として表記している。

- 推奨セキュリティ型（上記「変更前」に同じ）
  GnuTLSPriorities （略）:+VERS-TLS1.2:+VERS-TLS1.3:（略）

- 推奨セキュリティ型（TLS1.3 無効、上記「変更後」に同じ）
  GnuTLSPriorities （略）:+VERS-TLS1.2:（略）

- 高セキュリティ型
  GnuTLSPriorities （略）:+VERS-TLS1.2:+VERS-TLS1.3:（略）

- セキュリティ例外型
  GnuTLSPriorities （略）:+VERS-TLS1.0:+VERS-TLS1.1:+VERS-TLS1.2:+VERS-TLS1.3:（略）

- セキュリティ例外型（TLS1.0 無効）
  GnuTLSPriorities （略）:+VERS-TLS1.1:+VERS-TLS1.2:+VERS-TLS1.3:（略）

- セキュリティ例外型（TLS1.0、TLS1.1 無効）
  GnuTLSPriorities （略）:+VERS-TLS1.2:+VERS-TLS1.3:（略）

- セキュリティ例外型（TLS1.3 無効）
  GnuTLSPriorities （略）:+VERS-TLS1.0:+VERS-TLS1.1:+VERS-TLS1.2:（略）

3. 暗号スイート順序サーバ優先設定方法例

3.1. Apache+mod_ssl の場合

サーバ側の暗号スイート優先順位を使用するための設定は、1.1 節の暗号スイート順序サーバ優先設定の部分に以下の内容を記述する。

```
SSLHonorCipherOrder On
```

3.2. lighttpd の場合

サーバ側の暗号スイート優先順位を使用するための設定は、1.2 節の暗号スイート順序サーバ優先設定の部分に以下の内容を記述する。

```
ssl.honor-cipher-order = "enable"
```

3.3. nginx の場合

サーバ側の暗号スイート優先順位を使用するための設定は、1.3 節の暗号スイート順序サーバ優先設定の部分に以下の内容を記述する。

```
ssl_prefer_server_ciphers on;
```

3.4. Apache+mod_gnutls の場合

Apache+mod_gnutls では GnuTLSPriorities に指定する暗号スイート等の設定中に暗号スイート順序サーバ優先に関する設定が含まれる。サーバ側の暗号スイート優先順位を使用するためには、%SERVER_PRECEDENCE を指定する。

例えば、以下のような GnuTLSPriorities の指定に対してサーバ側の暗号スイート優先順位を使

用するよう設定するには、以下のように変更する。（太字部が変更箇所）

（変更前）
GnuTLSPriorities　NONE:+VERS-TLS1.2:+VERS-TLS1.3:+SIGN-ALL:+COMP-NULL:+GROUP-ALL:+ECDHE-ECDSA:+ECDHE-RSA:+DHE-RSA:+AES-128-GCM:+AES-256-GCM:+CHACHA20-POLY1305:+AES-128-CCM:+AES-256-CCM:+AES-128-CCM-8:+AES-256-CCM-8:+CAMELLIA-128-GCM:+CAMELLIA-256-GCM:+AEAD:+SHA384:+SHA256

（変更後）
GnuTLSPriorities　NONE:**%SERVER_PRECEDENCE**:+VERS-TLS1.2:+VERS-TLS1.3:+SIGN-ALL:+COMP-NULL:+GROUP-ALL:+ECDHE-ECDSA:+ECDHE-RSA:+DHE-RSA:+AES-128-GCM:+AES-256-GCM:+CHACHA20-POLY1305:+AES-128-CCM:+AES-256-CCM:+AES-128-CCM-8:+AES-256-CCM-8:+CAMELLIA-128-GCM:+CAMELLIA-256-GCM:+AEAD:+SHA384:+SHA256

4. 鍵交換パラメータの設定方法例

本章では DHE、ECDHE 鍵交換アルゴリズムで使用されるグループの指定方法について解説する。なお、RSA 鍵交換の際に使用される秘密鍵は証明書によるものであるため、本章では触れない。

4.1. OpenSSL による DHE パラメータファイルの生成

openssl コマンドにより、独自の 2048 ビットの位数を持つグループを指定する DHE パラメータファイルを PEM 形式で生成するには以下を実行する。

```
openssl dhparam -out dh2048.pem -outform PEM 2048
```

また、ECDHE パラメータファイルを PEM 形式で出力する方法もあるが、本ドキュメントでは 4.3 節以降に示すようにグループの名称を使用するため、使用しない。

4.2. GnuTLS による DHE パラメータファイルの生成

certtool コマンドにより、RFC7919 に掲載されたパラメータを PEM 形式で取得するには以下を実行する。この例では 2048 ビットの位数を持つグループを指定して取得している。

```
certtool --get-dh-params --bits=2048 --outfile=dh2048.pem
```

独自のグループを指定する DHE パラメータファイルを PEM 形式で生成することも可能であるが、GnuTLS では推奨されていない。この方法を使用する場合、例えば 2048 ビットの位数を持つ

グループであれば以下のコマンドを実行する。

```
certtool --generate-dh-params --bits=2048 --outfile=dh2048.pem
```

4.3. Apache+mod_ssl における DHE、ECDHE 鍵交換パラメータ設定

SSLCertificateFile は設定ファイル中で複数の指定が可能なプロパティであり、通常は PEM 形式の SSL サーバ証明書を指定するためのものである。

Apache 2.4.7 以降では、SSLCertificateFile で設定するファイルの中に、DHE 鍵交換で使用するグループを示すパラメータファイルを明示的に含めることができる。そのために、1.1 節の証明書と鍵の設定の部分で指定するファイル（1.1 節の例では/etc/ssl/chain.crt）に対して、4.1 節で生成したパラメータファイルを連結したファイルを新たに作成する。

- DHE 鍵交換で使用するグループのパラメータファイルによる指定例
  〔サーバ証明書とパラメータファイルを連結したファイル（chain-dh2048.crt）の作成（Linux 等）〕
  cat /etc/ssl/chain.crt dh2048.pem > /etc/ssl/chain-dh2048.crt
  〔httpd-ssl.conf での設定〕
  SSLCertificateFile /etc/ssl/chain-dh2048.crt

ECDHE 鍵交換で使用するグループの設定は、1.1 節の証明書と鍵の設定の部分に、以下のように追加する。

- ECDHE 鍵交換で使用するグループの指定例
  SSLOpenSSLConfCmd Groups "X25519:X448:P-256:P-384:P-521"

4.4. lighttpd における DHE、ECDHE 鍵交換パラメータ設定

lighttpd では、4.1 節で生成したパラメータファイルについて、1.2 節の証明書と鍵の設定の部分に、以下のように設定する。

- DHE 鍵交換で使用するグループのパラメータファイルによる指定例
  ssl.dh-file = "/etc/ssl/dh2048.pem"

- ECDHE 鍵交換で使用するグループの指定例
  ssl.openssl.ssl-conf-cmd = ("Groups" => "X25519:X448:P-256:P-384:P-521")

4.5. nginx における DHE、ECDHE 鍵交換パラメータ設定

nginx では、4.1 節で生成したパラメータファイルについて、1.3 節の証明書と鍵の設定の部分に、以下のように設定する。

- DHE 鍵交換で使用するグループのパラメータファイルによる指定例
  ssl_dhparam /etc/ssl/dh2048.pem;

- ECDHE 鍵交換で使用するグループの指定例
  ssl_ecdh_curve X25519:X448:P-256:P-384:P-521;

4.6. Apache+mod_gnutls における DHE、ECDHE 鍵交換パラメータ設定

Apache+mod_gnutls では、GnuTLSPriorities に指定する暗号スイート等の設定中にグループに関する設定が含まれる。鍵交換に使用されるグループを指定するためには、GROUP で始まる鍵交換で使用するグループの設定に関する部分を変更する。

- DHE パラメータファイルによる指定例
  GnuTLSPrioritiesde に記述するグループの設定には GROUP-EC-ALL を指定し、別途 GnuTLSDHFile の指定を追加する。
  例えば、高セキュリティ型で 2048 ビットの位数を持つグループを指定する DHE パラメータファイルを使用する場合は、以下のように指定する。（太字部が変更箇所）

  GnuTLSDHFile "/etc/ssl/dh2048.pem"
  GnuTLSPriorities NONE:%SERVER_PRECEDENCE:+VERS-TLS1.2:+VERS-TLS1.3:+SIGN-ALL:+COMP-NULL:**+GROUP-EC-ALL**:+ECDHE-ECDSA:+ECDHE-RSA:+DHE-RSA:+AES-128-GCM:+AES-256-GCM:+CHACHA20-POLY1305:+AES-128-CCM:+AES-256-CCM:+AES-128-CCM-8:+AES-256-CCM-8:+CAMELLIA-128-GCM:+CAMELLIA-256-GCM:+AEAD:+SHA384:+SHA256

- RFC7919 に掲載された DHE グループの指定例
  GnuTLSPrioritiesde に記述するグループの設定に、GROUP-EC-ALL と共に、GROUP-FFDHE2048、GROUP-FFDHE3072、GROUP-FFDHE4096、GROUP-FFDHE6144、GROUP-FFDHE8192 のいずれか、もしくは複数を指定する。
  例えば、高セキュリティ型で位数が 4096、6144、8192 ビットの各グループを使用する場合は以下のように指定する。（太字部が変更箇所）

  GnuTLSPriorities NONE:%SERVER_PRECEDENCE:+VERS-TLS1.2:+VERS-TLS1.3:+SIGN-ALL:+COMP-NULL:**+GROUP-EC-ALL**:**+GROUP-FFDHE4096**:**+GROUP-FFDHE6144**:**+GROUP-FFDHE8192**:+ECDHE-ECDSA:+ECDHE-RSA:+DHE-RSA:+AES-128-GCM:+AES-256-GCM:+CHACHA20-POLY1305:+AES-128-CCM:+AES-256-CCM:+AES-128-CCM-8:+AES-256-CCM-8:+CAMELLIA-128-GCM:+CAMELLIA-256-GCM:+AEAD:+SHA384:+SHA256

- ECDHE 鍵交換で使用するグループの指定例
  GnuTLSPrioritiesde に記述するグループの設定に、GROUP-DH-ALL と共に、GROUP-SECP256R1、GROUP-SECP384R1、GROUP-SECP521R1、GROUP-X25519 のいずれか、もしくは複数を指定する。
  例えば、高セキュリティ型で P-256、P-384、P-521、X25519 の各グループを使用する場合は以下のように指定する。（太字部が変更箇所）

  GnuTLSPriorities NONE:%SERVER_PRECEDENCE:+VERS-TLS1.2:+VERS-TLS1.3:+SIGN-ALL:+COMP-NULL:**+GROUP-DH-ALL**:**+GROUP-SECP256R1**:**+GROUP-SECP384R1**:**+GROUP-SECP521R1**:**+GROUP-X25519**:+ECDHE-ECDSA:+ECDHE-RSA:+DHE-RSA:+AES-128-GCM:+AES-256-GCM:+CHACHA20-POLY1305:+AES-128-CCM:+AES-256-CCM:+AES-128-CCM-8:+AES-256-CCM-8:+CAMELLIA-128-GCM:+CAMELLIA-256-GCM:+AEAD:+SHA384:+SHA256

5. HTTP Strict Transport Security（HSTS）の設定方法例

5.1. Apache の場合

HTTP ヘッダに HSTS の情報を追加するために、設定ファイルに以下の記述を追加する。なお、max-age は有効期間を表し、この例では 365 日（31,536,000 秒）の有効期間を設定することを意味している。また、includeSubDomains がある場合、サブドメインにも適用される。

```
Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"
```

なお、HTTP によるアクセスを全て HTTPS にリダイレクトするためには、<VirtualHost *:80>中に以下のような RewriteRule、RewriteEngine の設定を追加する。

```
LoadModule rewrite_module modules/mod_rewrite.so
<VirtualHost *:80>
        （中略）
    ServerAlias *
    RewriteEngine On
    RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [redirect=301]
</VirtualHost>
```

5.2. lighttpd の場合

HTTP ヘッダに HSTS の情報を追加するために、設定ファイル（1.2 節の場合、modules.conf と lighttpd.conf）に以下の記述を追加する。なお、max-age は有効期間を表し、この例では 365 日

（31,536,000 秒）の有効期間を設定することを意味している。また、includeSubDomains がある場合、サブドメインにも適用される。

〔modules.conf での設定〕
```
server.modules = (
        （中略）
        "mod_setenv"
)
```

〔lighttpd.conf での設定〕
```
setenv.add-response-header = (
        "Strict-Transport-Security" => "max-age=31536000; includeSubDomains"
)
```

なお、HTTP によるアクセスを全て HTTPS にリダイレクトするためには、設定ファイル（1.2 節の場合、modules.conf と lighttpd.conf）に以下のような設定を追加する。

〔modules.conf での設定〕
```
server.modules = (
        （中略）
        "mod_redirect"
)
```

〔lighttpd.conf での設定〕
```
$HTTP["scheme"] == "http" {
        （中略）
        $HTTP["host"] =~ ".*" {
                url.redirect = (".*" => "https://%0$0")
        }
}
```

5.3. nginx の場合

HTTP ヘッダに HSTS の情報を追加するために、設定ファイルに以下の記述を追加する。なお、max-age は有効期間を表し、この例では 365 日（31,536,000 秒）の有効期間を設定することを意味している。また、includeSubDomains がある場合、サブドメインにも適用される。

```
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains";
```

なお、HTTP によるアクセスを全て HTTPS にリダイレクトするためには、"listen 80;"を含む server 中に、以下のような設定を追加する。

```
server {
        listen 80;
        （中略）
        return 301 https://$host$request_uri;
}
```

6. OCSP stapling の設定方法例

6.1. Apache+mod_ssl の場合

OCSP stapling を有効にするために、設定ファイルに以下の記述を追加する。

なお、この例における SSLStaplingCache の末尾の括弧内はキャッシュサイズを表し、この例では 32,768 バイトを設定することを意味している。また、<VirtualHost *:443>の外側に記載すること。

```
SSLStaplingCache "shmcb:/var/run/apache2/stapling_cache(32768)"
<VirtualHost *:443>
        （中略）
        SSLUseStapling on
</VirtualHost>
```

6.2. nginx の場合

OCSP stapling を有効にするために、設定ファイルに以下の記述を追加する。

ここで ssl_trusted_certificate に指定する証明書は OCSP レスポンダからの応答の検証のみならず、クライアント認証を行う際のクライアント証明書の検証にも使用される。OCSP stapling とクライアント認証を同時に有効にしないなど、予期しないクライアント証明書を受け入れることが無いよう注意すること。

```
server {
        （中略）
        ssl_stapling on;
        ssl_stapling_verify on;
        ssl_trusted_certificate /etc/ssl/ca-certs.pem;
}
```

6.3. Apache+mod_gnutls の場合

OCSP stapling を有効にするために、設定ファイルに以下の記述を追加する。

なお、この例における GnuTLSOCSPCache の末尾の括弧内はキャッシュサイズを表し、この例では 32,768 バイトを設定することを意味している。また、<VirtualHost *:443>の外側に記載すること。

ただし mod_gnutls の OCSP stapling 実装には不具合や互換性の問題が存在するため、証明書の内容や OCSP レスポンダの実装によっては OCSP レスポンダから正常な応答を得られず、クライアントに証明書の状態を送信できないことがある点に留意すること。

```
GnuTLSOCSPCache "shmcb:/var/run/apache2/stapling_cache(32768)"

<VirtualHost *:443>
    GnuTLSOCSPStapling On
</VirtualHost>
```

7. 設定内容の確認方法

7.1. オンラインサービスを使用した確認方法

Qualys, Inc.が SSL Labs 内で提供する SSL Server Test を使用してインターネットに接続されたサーバの設定を確認することができる。結果を秘匿したい場合は、「Do not show the results on the boards」にチェックを入れる必要がある点に注意する。

SSL Server Test
https://www.ssllabs.com/ssltest/

DHE、ECDHE 鍵交換で使用されるグループは Configuration→Cipher Suites にて確認できる。図 1 の例ではサーバは ECDHE では P-256、DHE では 2048 ビットの位数を持つグループを使用している。

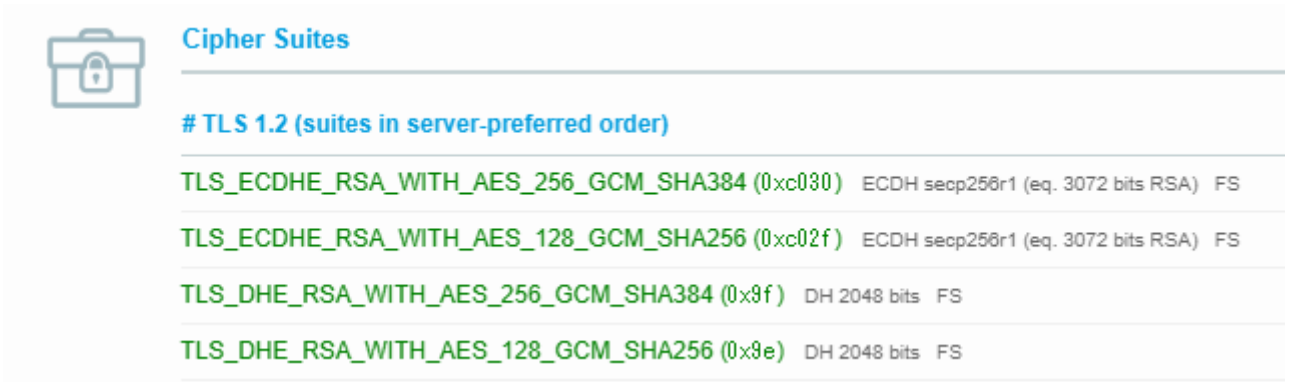

図 1　Cipher Suites の表示(ssllabs.com より引用)

ECDHE 鍵交換でサポートされているすべてのグループは Configuration→Protocol Details→Supported Named Groups にて確認できる。図 2 の例ではサーバは P-256、P-521、P-384、secp256k1

の 4 種類のグループをサポートしている。

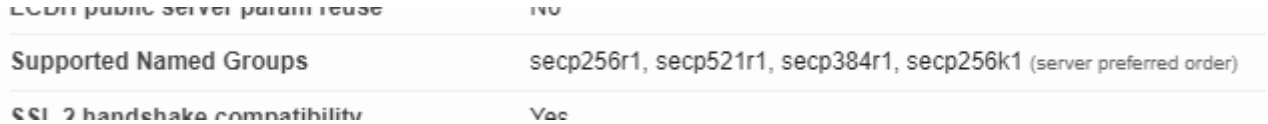

図 2　Supported Named Groups の表示(ssllabs.com より引用)

7.2.　コマンドラインツールを使用した確認方法

フリーソフトウェア testssl.sh を用いて Linux 等のコマンドラインからサーバの設定を確認することができる。

testssl.sh
https://testssl.sh/
https://github.com/drwetter/testssl.sh

（実行例）
./testssl.sh example.jp:443

DHE 鍵交換で使用されるグループは Testing robust (perfect) forward secrecy→Finite field group で確認することができる。

ECDHE 鍵交換で使用されるグループは Testing robust (perfect) forward secrecy→Elliptic curves offered で確認することができる。

（出力例・抜粋）

Elliptic curves offered:	prime256v1 secp384r1 secp521r1 X25519 X448
DH group offered:	RFC3526/Oakley Group 14 (2048 bits)

7.3.　openssl コマンドを用いた確認方法

OpenSSL を用いてサーバの鍵交換パラメータを確認するには、openssl s_client を使用する。

- TLS1.2 以前での DHE 鍵交換で使用されるグループの確認方法

（実行例）
openssl s_client -cipher DHE -tls1_2 -connect example.jp:443

（出力例・抜粋）
この例ではサーバは DHE 鍵交換に 2048 ビットの位数を持つグループを使用している。
Server Temp Key: DH, 2048 bits

- TLS1.2 以前での ECDHE 鍵交換で使用されるグループの確認方法

（実行例）
openssl s_client -cipher ECDHE -tls1_2 -connect example.jp:443

（出力例・抜粋）
この例ではサーバは X25519 を使用している。
Server Temp Key: X25519, 253 bits

- TLS1.3 での鍵交換で使用されるグループの確認方法

（実行例）
openssl s_client -tls1_3 -connect example.jp:443

（出力例・抜粋）
この例ではサーバは X25519 を使用している。
Server Temp Key: X25519, 253 bits

- RSA 証明書の鍵ビット数の確認方法（この鍵は TLS1.2 以前の RSA 鍵交換でも使用される）

（実行例）
openssl s_client -cipher DHE+aRSA:kRSA -tls1_2 -connect example.jp:443 | openssl x509 -text

（出力例・抜粋）
この例ではサーバは、公開鍵が 2048 ビットの RSA 証明書を使用している。

```
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
        Public-Key: (2048 bit)
```

- ECDSA 証明書の鍵ビット数とグループの確認方法

（実行例）
openssl s_client -cipher ECDSA -tls1_2 -connect example.jp:443 | openssl x509 -text

（出力例・抜粋）
この例ではサーバは、P-256 をグループに使用した ECDSA 証明書を使用している。

```
Subject Public Key Info:
```

Public Key Algorithm: id-ecPublicKey
Public-Key: (256 bit)
pub: （略）
ASN1 OID: prime256v1
NIST CURVE: P-256

7.4. ブラウザを用いた確認方法

ウェブブラウザソフトウェアを用いて鍵交換で使用されたグループを確認することもできる。以下に主要なブラウザでの確認方法を示す。

本節は以下のソフトウェアバージョンを参考に作成された。

Google Chrome 79.0.3945.117
Safari 13.0.5 (15608.5.11)
Mozilla Firefox 72.0.1
Internet Explorer 11.592.18362.0 （更新バージョン 11.0.170）

- Google Chrome
  ウェブページを開いた状態で F12 キーを押下して表示される DevTools の Security タブで、以下の要素を確認することができる。ただし、現在の Google Chrome は DHE 鍵交換をサポートしていない点に留意する。
  プロトコルバージョン
  鍵交換アルゴリズム
  鍵交換で使用されたグループ
  暗号アルゴリズム・暗号利用モード

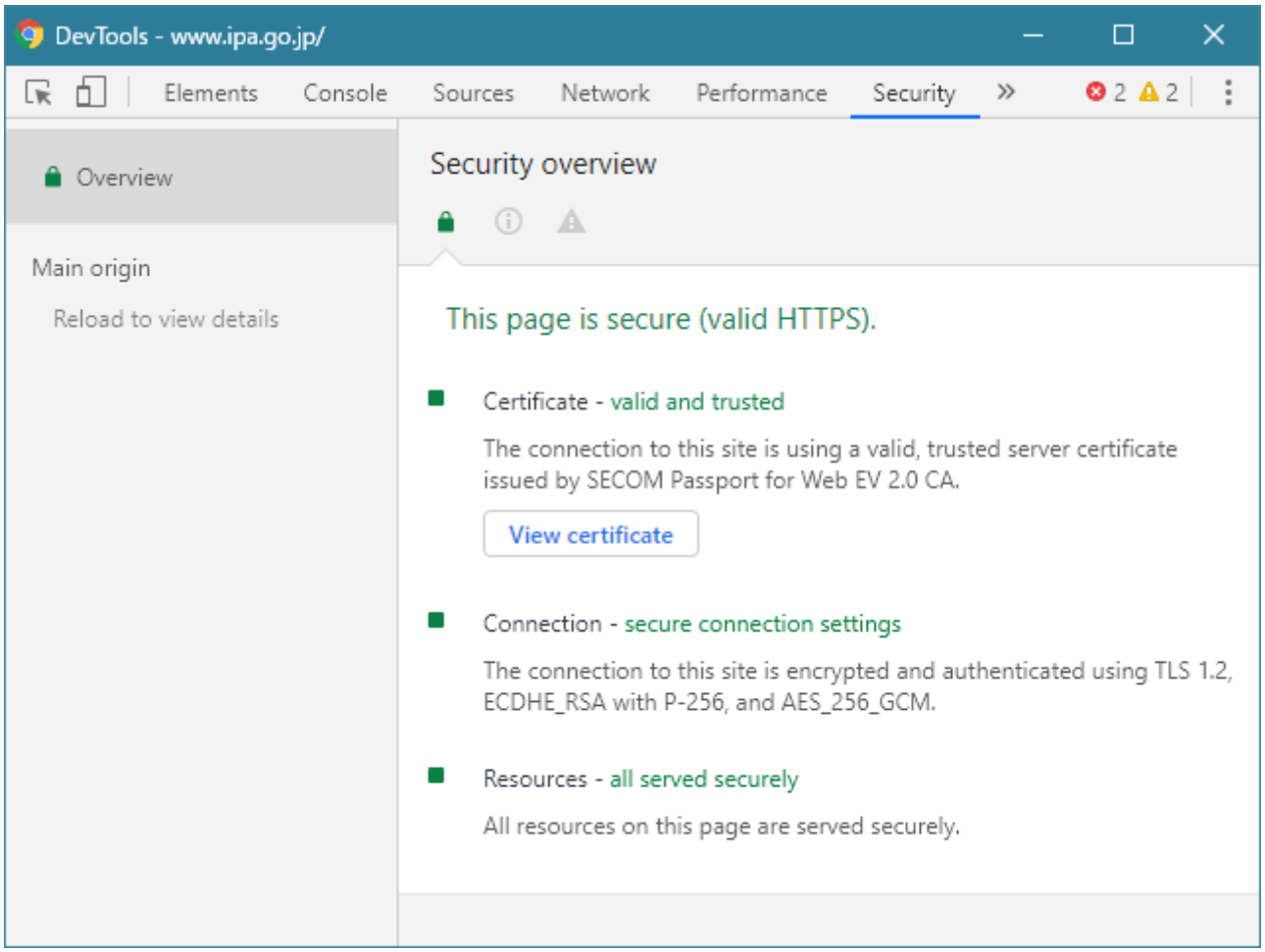

図 3　Google Chrome における表示

- Safari

  「開発」メニューの「Web インスペクタを表示」を選択することで表示される「Web インスペクタ」ウィンドウの「ネットワーク」タブで表示される各リソースの「セキュリティ」タブで、以下の要素を確認することができる。(「情報がありません。」となった場合は再読み込みを行う)

  プロトコルバージョン

  暗号スイート

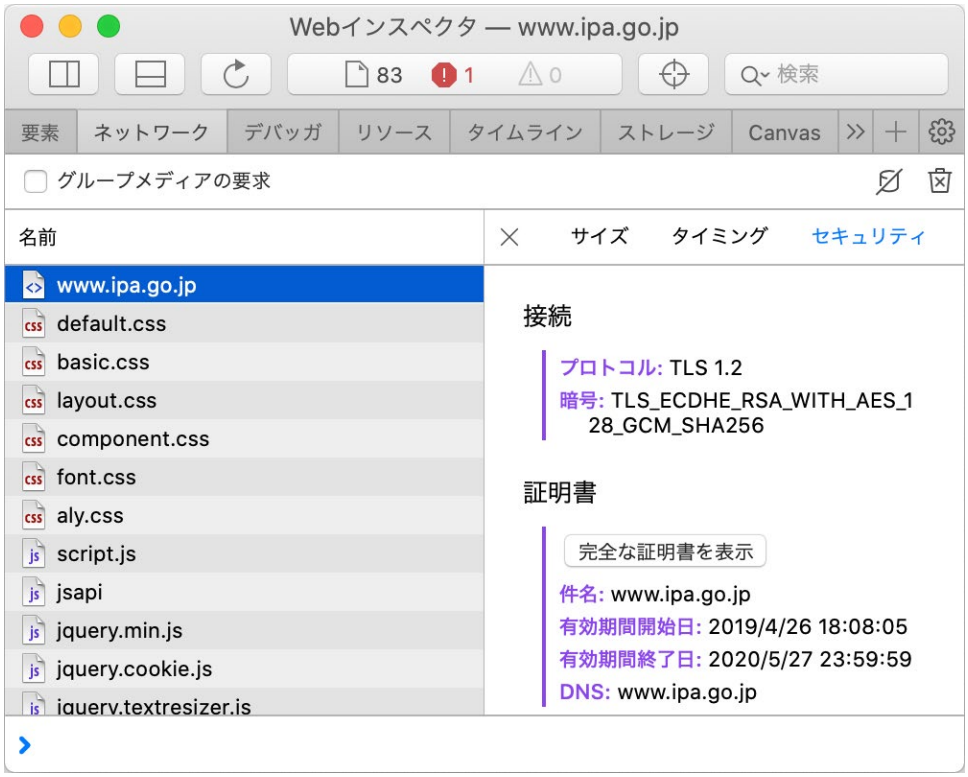

図 4　Safari における表示

- Mozilla Firefox
  ウェブページを開いた状態でコンテキストメニューの「ページの情報を表示(I)」を選択することで表示される「ページ情報」ダイアログの「セキュリティ(S)」タブで、以下の要素を確認することができる。
  プロトコルバージョン
  暗号スイート

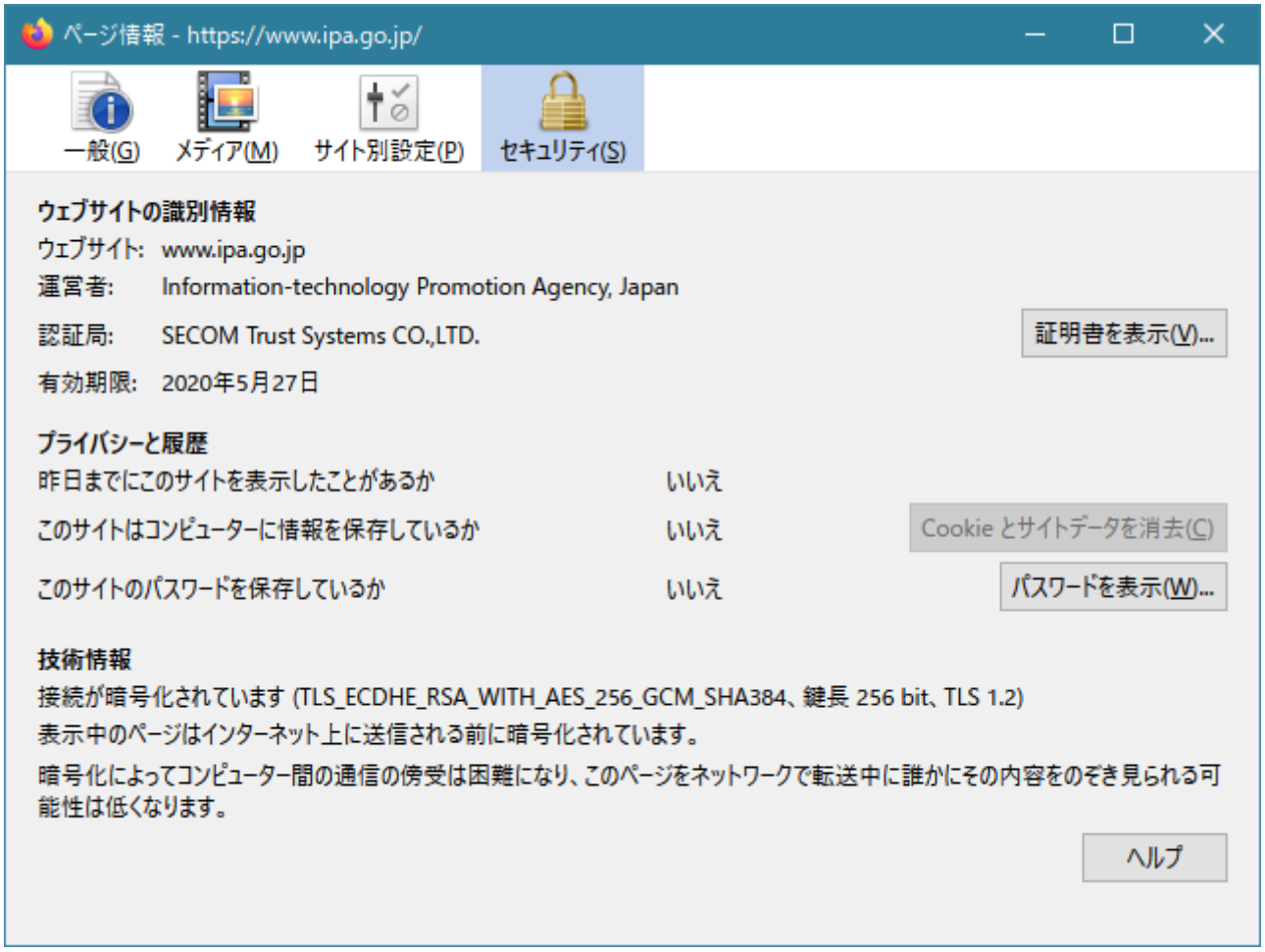

図 5　Mozilla Firefox における表示

- Internet Explorer

  ウェブページを開いた状態でコンテキストメニューの「プロパティ(P)」を選択することで表示される「プロパティ」ダイアログで、以下の要素を確認することができる。

  プロトコルバージョン
  暗号アルゴリズム
  鍵交換アルゴリズム
  鍵交換で使用されたグループの位数のビット数

図 6　Internet Explorer における表示

8. 修正履歴

- 2021.12.02 (ver 1.1)

  「4.3. Apache+mod_ssl における DHE、ECDHE 鍵交換パラメータ設定」での説明文の修正

  - 「パラメータファイルを追記する。」→「パラメータファイルを連結したファイルを作成する。」
  - 「サーバ証明書への追記（Linux 等）」→「サーバ証明書とパラメータファイルを連結したファイル(chain-dh2048.crt)の作成（Linux 等）」