TLS 暗号設定 サーバ設定編

Ver 2.0

令和 6 年 6 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。採用したバージョンについて、OpenSSL は 2024 年 4 月 1 日時点で最新の LongTermSupport 版[1]とし、それ以外のソフトウェアは同時点の最新安定版とした。

OpenSSL 3.0.13
GnuTLS 3.7.10
Apache httpd 2.4.58
lighttpd 1.4.75
nginx 1.24.0

なお、mod_gnutls は 2023 年 10 月にリリースされた Apache 2.4.58 ではサポートが廃止されたため、本書の対象ソフトウェアから mod_gnutls を削除した。mod_gnutls での設定例を確認する場合はアーカイブの Ver.1.1 を参考にされたい。

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。本節は Ver.1.1 からの変更はない。

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

    # 証明書と鍵の設定[2]
    SSLCertificateFile /etc/ssl/chain.crt
    SSLCertificateKeyFile /etc/ssl/server.key
```

[1] サポート期限が 2026 年 9 月 7 日までと安定版の中で最も長い。
[2] 設定する内容は以下の通り
/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

```
# 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

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。公式に推奨されている TLS1.2 での暗号スイート設定方法が変更されたため、本節では Ver.1.1 からコマンドと説明を追加した。

```
$SERVER["socket"] == "0.0.0.0:443" {
    ssl.engine = "enable"
    （中略）
    # モジュールの追加
    server.modules += ( "mod_openssl" )

    # 証明書と鍵の設定[3]
    ssl.pemfile = "/etc/ssl/serverkey_cert.pem"
    ssl.privkey = "/etc/ssl/serverkey_cert.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
    ssl.openssl.ssl-conf-cmd += ("CipherString" => "TLS1.2 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")
```

[3] 設定する内容は以下の通り
/etc/ssl/serverkey_cert.pem：サーバ証明書および中間証明書
/etc/ssl/serverkey_cert.key：サーバ証明書に対応する秘密鍵

```
# 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)
# 暗号スイート順序サーバ優先設定
ssl.openssl.ssl-conf-cmd += (" Options" =>"+ServerPreference")
```

```
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
```

```
}
```

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。nginx が TLS1.3 に対応したため、本節では Ver.1.1 からコマンドを追加した。

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

    # 証明書と鍵の設定[4]
    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
    ssl_ciphers "TLS1.2 以前用暗号スイート設定文字列";
    ssl_conf_command Ciphersuites "TLS1.3 暗号スイート用文字列";

    # 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;
```

[4] 設定する内容は以下の通り
/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 章以降を参照のこと

}

2. プロトコルバージョンの設定方法例

本章では Ver.1.1 では記載されていなかった「高セキュリティ型（TLS1.2 無効）」の設定例を追加した。

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（TLS1.2 無効）
  SSLProtocol TLSv1.3

- 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（TLS1.2 無効）
  ssl.openssl.ssl-conf-cmd += ("MinProtocol" => "TLSv1.3", "MaxProtocol" => "TLSv1.3")

- 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（TLS1.2 無効）
  ssl_protocols TLSv1.3;

- 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;

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。本節は Ver.1.1 からの変更はない。

```
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。公式で推奨されている設定方法が変更されたため、本節は Ver.1.1 からコマンドを変更した。

```
ssl.openssl.ssl-conf-cmd = ("Options" => "+ServerPreference")
```

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。本節は Ver.1.1 からの変更はない。

```
ssl_prefer_server_ciphers on;
```

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。本節は Ver.1.1 からの変更はない。

```
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、本書では 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。本節は Ver.1.1 からの変更はない。

```
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 は通常 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。本節は Ver.1.1 からの変更はない。

- 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。公式で推奨されている設定方法が変更されたため、本節では Ver.1.1 からコマンドを変更した。

- 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.openssl.ssl-conf-cmd += ("DHParameters" => " /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。本節は Ver.1.1 からの変更はない。

- 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;

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

本章は Ver.1.1 からの変更はない。

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。本節では Ver.1.1 から SSLStaplingStandardCacheTimeout の設定を追加した。

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、SSLStaplingStandardCacheTimeout の末尾の値は有効な OCSP 応答がキャッシュから失効するまでの秒数を表し、この例では 3600 秒の有効期間を設定することを意味している。なお、3600 はデフォルト値であり、デフォルト値をそのまま使う場合には省略可能である。この記述は<VirtualHost *:443>の外側に記載すること。

```
SSLStaplingCache "shmcb:/var/run/apache2/stapling_cache(32768)"
SSLStaplingStandardCacheTimeout 3600

<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。本節は Ver.1.1 からの変更はない。

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

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。本節は Ver.1.1 からの変更はない。

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。

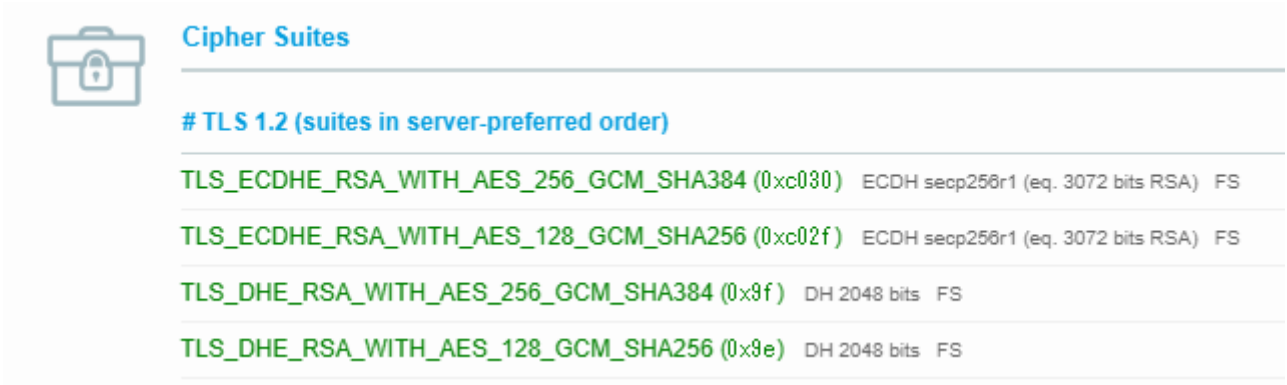

図 1　Cipher Suites の表示(ssllabs.com より引用)

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。

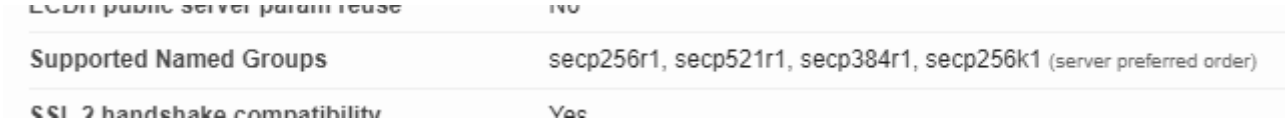

図 2　Supported Named Groups の表示(ssllabs.com より引用)

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。本節は Ver.1.1 からの変更はない。

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。本節は Ver.1.1 からの変更はない。

- 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 Ver.1.1 からソフトウェアバージョン、確認方法、表示例を変更した。

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 123.0.6312.123
Safari 17.4.1 (19618.1.15.11.14)
Mozilla Firefox 124.0.2
Microsoft Edge 123.0.2420.97

- 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

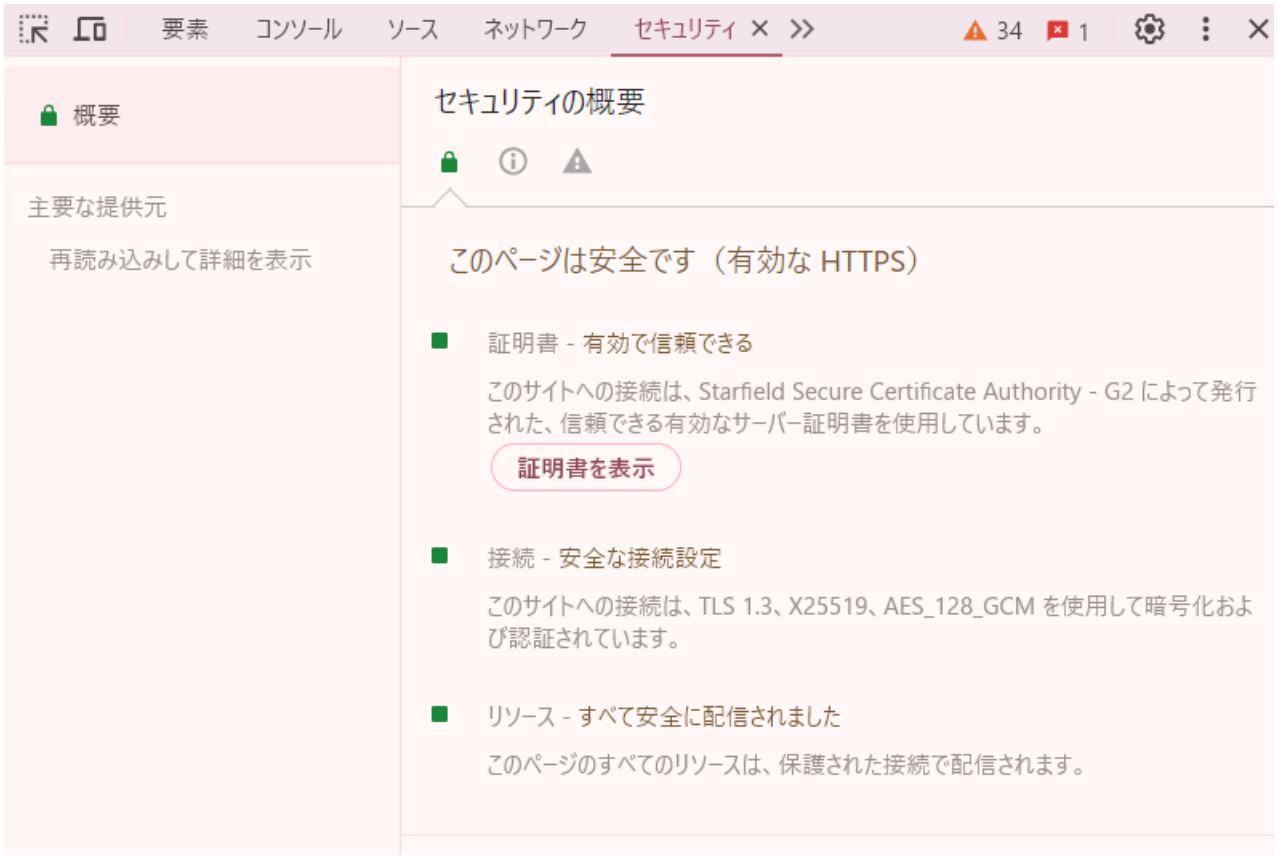

図 3　Google Chrome における表示

- 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

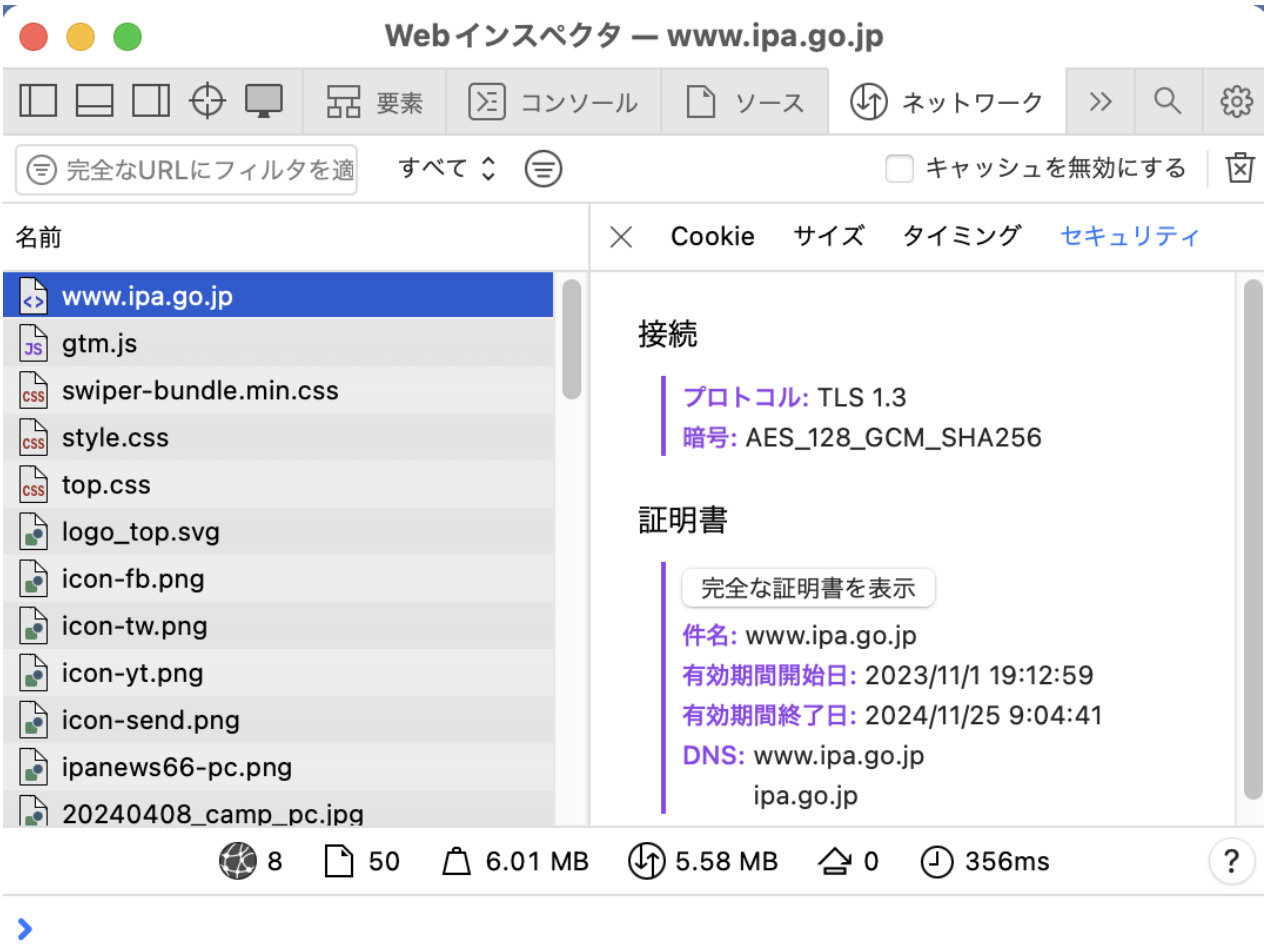

図 4　Safari における表示

- Mozilla Firefox

  ウェブページを開いた状態で Ctrl キー+I キーを押下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

- Microsoft Edge

  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「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。

  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 6　Microsoft Edge における表示

8. 修正履歴

修正日	修正内容
2020.7.7（Ver.1.0）	初版発行
2021.12.2（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）」
2024.6.17（Ver.2.0）	● 最新安定版のバージョンのライブラリ、ソフトウェアを利用した内容に更新 ● mod_gnutls の公式メンテナンスが廃止されたため、対象ソフトウェアから mod_gnutls を削除。関連する節を削除 ● 「7.4. ブラウザを用いた確認方法」での Internet Explorer を Microsoft Edge に変更