SSL/TLS 暗号設定 サーバ設定編

平成 27 年 8 月

独立行政法人 情報処理推進機構
国立研究開発法人 情報通信研究機構

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。

1. サーバ設定方法例のまとめ

1.1. Apache 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 2 章も参照のこと
        SSLCipherSuite "暗号スイート設定"

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、Public Key Pinning の設定をする場合には、
ここに追記する。ガイドライン 7.2 節及び 本書 4 章以降も参照のこと

</VirtualHost>
```

[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

1.2. lighttpd の場合

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

〔modules.conf での設定〕

```
server.modules = (
        （中略）
       "mod_setenv"
)
```

〔lighttpd.conf での設定〕

```
$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"
       ssl.ca-file = "/etc/ssl/ca.crt"
```

暗号スイート設定。暗号スイートの設定例 2 章も参照のこと

```
       ssl.cipher-list = "暗号スイート設定"
```

プロトコルバージョン設定。2.2 節も参照のこと

```
       ssl.use-プロトコルバージョン  = "利用可否"
```

暗号スイート順序サーバ優先設定

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

HTTP Strict Transport Security、Public Key Pinning の設定をする場合には、ここに追記する。ガイドライン 7.2 節及び本書 4 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。

```
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 2 章も参照のこと

```
ssl_ciphers "暗号スイート設定";
```

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、Public Key Pinning の設定をする場合には、ここに追記する。ガイドライン 7.2 節及び本書 4 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

2.1. Apache の場合

Apache での設定例を以下に示す。

- 高セキュリティ型
  SSLProtocol TLSv1.2

- 推奨セキュリティ型
  SSLProtocol All -SSLv2 -SSLv3

- セキュリティ例外型
  SSLProtocol All -SSLv2

2.2. lighttpd の場合

lighttpd での設定例を以下に示す。

- 高セキュリティ型
  ssl.use-tlsv1.1 = "disable"
  ssl.use-tlsv1 = "disable"
  ssl.use-sslv3 = "disable"
  ssl.use-sslv2 = "disable"

- 推奨セキュリティ型
  ssl.use-sslv3 = "disable"
  ssl.use-sslv2 = "disable"

- セキュリティ例外型
  ssl.use-sslv2 = "disable"

2.3. nginx の場合

nginx での設定例を以下に示す。なお、TLS1.1 及び TLS1.2 は、バージョンが 1.1.13 または 1.0.12 であり、かつ OpenSSL のバージョンが 1.0.1 以上の時に利用できる。

- 高セキュリティ型（Ver. 1.1.13/1.0.12 かつ OpenSSL ver. 1.0.1 以上）
  ssl_protocols TLSv1.2;

- 推奨セキュリティ型
  ssl_protocols TLSv1.2 TLSv1.1 TLSv1;（Ver. 1.1.13/1.0.12 かつ OpenSSL ver. 1.0.1 以上）
  ssl_protocols TLSv1;

- セキュリティ例外型
  ssl_protocols TLSv1.2 TLSv1.1 TLSv1 SSLv3;（Ver. 1.1.13/1.0.12 かつ OpenSSL ver. 1.0.1 以上）
  ssl_protocols TLSv1 SSLv3;

2.4. Microsoft IIS の場合

各 OS におけるプロトコルバージョンのサポート状況は以下の通りである。

	TLS1.2	TLS1.1	TLS1.0	SSL3.0	SSL2.0
Windows Server 2008	×	×	○	○	○
Windows Vista	×	×	○	○	○
Windows Server 2008 R2（以降）	○	○	○	○	○
Windows 7 以降の Windows	○	○	○	○	○

凡例：○ サポートあり　　× サポートなし

サポートされているプロトコルバージョンの利用可否については、以下の設定例に従い、レジストリを設定する。

参考情報：
特定の暗号化アルゴリズムおよび Schannel.dll のプロトコルの使用を制限する方法
https://support.microsoft.com/en-us/kb/245030

- 高セキュリティ型
  HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Control¥SecurityProviders¥Schannel¥Protocols¥SSL 2.0¥Server
  "DisabledByDefault"=dword:00000001
  HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Control¥SecurityProviders¥Schannel¥Protocols¥SSL 3.0¥Server
  "DisabledByDefault"=dword:00000001
  HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Control¥SecurityProviders¥Schannel¥Protocols¥TLS 1.0¥Server
  "DisabledByDefault"=dword:00000001
  HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Control¥SecurityProviders¥Schannel¥Protocols¥TLS 1.1¥Server
  "DisabledByDefault"=dword:00000001

- 推奨セキュリティ型
  HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Control¥SecurityProviders¥Schannel¥Protocols¥SSL 2.0¥Server
  "DisabledByDefault"=dword:00000001
  HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Control¥SecurityProviders¥Schannel¥Protocols¥SSL 3.0¥Server
  "DisabledByDefault"=dword:00000001

- セキュリティ例外型
  HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Control¥SecurityProviders¥Schannel¥Protocols¥SSL 2.0¥Server
  "DisabledByDefault"=dword:00000001

3. 鍵パラメータファイルの設定方法例

3.1. OpenSSL による DHE、ECDH、ECDHE 鍵パラメータファイルの生成

OpenSSL コマンドにより、DHE 鍵パラメータファイル（2048 ビット）を生成するには以下を実行する。

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

また、ECDH、ECDHE 鍵パラメータファイル（256 ビット）は以下のようにして生成することができる。

```
openssl ecparam -out prime256v1.pem -name prime256v1
```

3.2. Apache における DHE、ECDH、ECDHE 鍵パラメータ設定

SSLCertificateFile は設定ファイル中でいくつも指定できるプロパティであり、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、ECDH、ECDHE の鍵長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 節の場合、/etc/ssl/chain.crt）に対して、3.1 節で生成した鍵パラメータファイルを追記する。

例えば、linux 等であれば以下の処理を行う。

- DHE 鍵パラメータファイル（2048 ビット）の指定例

  ```
  cat dh2048.pem >> /etc/ssl/chain.crt
  ```

- ECDH、ECDHE 鍵パラメータファイル（256 ビット）の指定例

  ```
  cat prime256v1.pem >> /etc/ssl/chain.crt
  ```

3.3. lighttpd における DHE、ECDH、ECDHE 鍵パラメータ設定

lighttpd では、 3.1 節で生成した鍵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、以下のように追加する。

- DHE の鍵パラメータファイル（2048 ビット）の指定例

  ```
  ssl.dh-file = "/etc/ssl/dh2048.pem"
  ```

- ECDH、ECDHE の楕円曲線パラメータ（256 ビット）の指定例

  ```
  ssl.ec-curve = "prime256v1"
  ```

3.4. nginx における DHE、ECDH、ECDHE 鍵パラメータ設定

nginx では、3.1 節で生成した鍵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、以下のように追加する。

- DHE の鍵パラメータファイル（2048 ビット）の指定例

```
ssl_dhparam /etc/ssl/dh2048.pem;
```

- ECDH、ECDHE の楕円曲線パラメータ（256 ビット）の指定例

```
ssl_ecdh_curve prime256v1;
```

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

4.1. 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 の場合に強制的に HTTPS にリダイレクトするためには、<VirtualHost *:80>中の RewriteRule、RewriteEngine の設定を以下のように追記する。

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

4.2. 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、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。

```
setenv.add-response-header = (
    "Strict-Transport-Security" => "max-age=31536000; includeSubDomains"
```

)

なお、HTTP の場合に強制的に 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）に以下のように追記する。

〔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")
        }
}
```

4.3. 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";
```

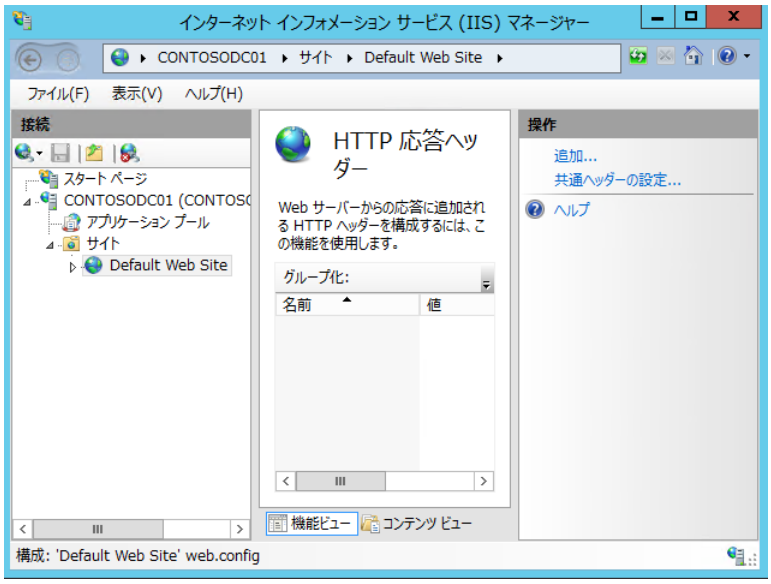
なお、HTTP の場合に強制的に HTTPS にリダイレクトするためには、"listen 80;"中に、以下のように追記する。

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

4.4. Microsoft IIS の場合

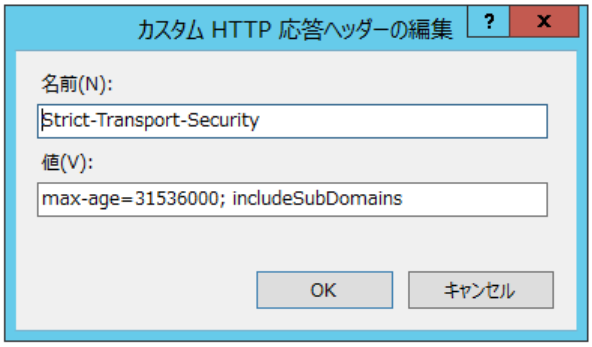
IIS では、HTTP ヘッダに HSTS の情報を追加するために、以下の手順により設定する。

1） 「IIS マネージャー」を開く
2） 「機能ビュー」を開く
3） 「HTTP 応答ヘッダ」をダブルクリックする
4） 「操作」のペインで「追加」をクリックする



5） 「名前」「値」の箇所を以下のように設定する。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

名前：Strict-Transport-Security
値　：max-age=31536000; includeSubDomains



6） 「OK」をクリックする。

5. OCSP Stapling の設定方法例

5.1. Apache 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、SSLStaplingCache の stapling_cache はキャッシュサイズを表し、この例では 128,000 バイトを設定することを意味している。また、<VirtualHost *:443>の前に記載すること。

```
SSLStaplingCache shmcb:/tmp/stapling_cache(128000)
<VirtualHost *:443>
        （中略）
        SSLCACertificateFile /etc/ssl/ca-certs.pem
        SSLUseStapling on
</VirtualHost>
```

5.2. 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。

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

5.3. Microsoft IIS の場合

Windows Server 2008 以降の Windows では、デフォルトで OCSP Stapling が設定されている。

6. Public Key Pinning の設定方法例

Public Key Pinning で使用される HTTP ヘッダの属性名は"Public-Key-Pins"であり、ヘッダの例は以下のようになる。

Public-Key-Pins 'pin-sha256="証明書の公開鍵情報の SHA-256 ハッシュ値（pinned fingerprint）の Base64 値"; pin-sha256="バックアップのための公開鍵情報の SHA-256 ハッシュ値（backup pinned fingerprint）の Base64 値"; max-age=有効期間; includeSubDomains'

- Pinned fingerprint は、エンドエンティティ（SSL サーバ証明書）から最上位の中間証明書までの検証チェーン中のいずれかの証明書の公開鍵情報のハッシュ値である。どの証明書を選んでもよいが、現時点ではハッシュ値を計算するハッシュ関数として SHA-256 のみが利用できる。また、複数の証明書を選択して併記することもできる。

- Backup pinned figerprint は、SSL サーバ証明書で使われている鍵ペア（primary key pair）が漏えい等の何らかの理由で利用できなくなったときに、サーバ運用者があらかじめ管理している予備の鍵ペア（secondary/backup key pair）に切り替えて暫定的に利用できるようするための公開鍵情報のハッシュ値である。エンドエンティティ（SSL サーバ証明書）から最上位の中間証明書までの検証チェーンには含まれない形の公開鍵情報として設定され、通常は利用しない。本来利用している鍵ペアが利用できなくなっても予備の鍵ペアを使うことでサイト運用をそのまま継続することができるが、予備の秘密鍵が漏えいした際の対策がなくなるので、予備の秘密鍵を厳重に管理することが必要である。
- max-age により有効期間（秒）を指定する。なお、Public Key Pinning が設定されたサイトで一度検証が OK になると max-age の期間内は Known Pinned Host として有効なサイトと判断される。このため、あまりに長い有効期間を設定するのは望ましくない。
- includeSubDomains がある場合、サブドメインにも適用される。

〔具体的な表記例〕

Public-Key-Pins 'pin-sha256="QtXc8+scL7K6HiPksQ8mqIyY08Xdc4Z5raHT+xSh9/s="; pin-sha256="kb6xLprt35abNnSn74my4Dkfya9arbk5zN5a60YzuqE="; max-age=3000; includeSubDomains'

この例では、エンドエンティティ（SSL サーバ証明書）から最上位の中間証明書までの検証チェーン中のいずれか 1 つの証明書の公開鍵情報の SHA-256 ハッシュ値（pinned fingerprint）の Base64 値が"QtXc8+"から始まる値であり、バックアップのための公開鍵情報の SHA-256 ハッシュ値（backup pinned fingerprint）の Base64 値が"kb6xLp"から始まる値であることを意味する。また、max-age は 50 分（3,000 秒）の有効期間を意味している。

なお、ハッシュ値の Base64 値を簡単に計算する方法はいくつかある。

例えば、OpenSSL を利用する方法や、PEM 形式のサーバ証明書を入力して Public-Key-Pins ヘッダを自動作成するサイト[2]がある。

〔OpenSSL を利用する方法〕

PEM 形式のあるサーバ証明書（certificate.pem）の SHA-256 ハッシュ値の Base64 値を求める場合は、以下のように計算する

```
openssl x509 -noout -in certificate.pem -pubkey | openssl asn1parse -noout -inform pem -out public.key;
openssl dgst -sha256 -binary public.key | openssl enc -base64
```

6.1. Apache の場合

6 章の表記に従い、mod_headers モジュールを有効にし、以下の設定を追加する。

[2] https://projects.dm.id.lv/s/pkp-online/calculator.html

Header always set Public-Key-Pins 'pin-sha256="証明書の公開鍵情報の SHA-256 ハッシュ値（pinned fingerprint）の Base64 値"; pin-sha256="バックアップのための公開鍵情報の SHA-256 ハッシュ値（backup pinned fingerprint）の Base64 値"; max-age=有効期間'

ちなみに、mod_headers モジュールを有効にするためには、httpd.conf において
LoadModule headers_module modules/mod_headers.so
を設定する。

6.2. lighttpd での設定例[3]

6 章の表記に従い、設定ファイルにおいて、以下の設定を追加する。

setenv.add-response-header = (
"Public-Key-Pins" => "pin-sha256=¥"証明書の公開鍵情報の SHA-256 ハッシュ値（pinned fingerprint）の Base64 値¥"; pin-sha256=¥"バックアップのための公開鍵情報の SHA-256 ハッシュ値（backup pinned fingerprint）の Base64 値¥"; max-age=有効期間",
)

6.3. nginx の場合

6 章の表記に従い、設定ファイルにおいて、以下の設定を追加する。

add_header Public-Key-Pins 'pin-sha256="証明書の公開鍵情報の SHA-256 ハッシュ値（pinned fingerprint）の Base64 値"; pin-sha256="バックアップのための公開鍵情報の SHA-256 ハッシュ値（backup pinned fingerprint）の Base64 値"; max-age=有効期間';

6.4. Microsoft IIS の場合

IIS では、6 章の表記に従い、以下の手順により設定する。

1） 「IIS マネージャー」を開く
2） 「機能ビュー」を開く
3） 「HTTP 応答ヘッダ」をダブルクリックする
4） 「操作」のペインで「追加」をクリックする
5） 6 章の表記に従い、「名前」「値」の箇所に以下のように設定する。この例では SHA-256 と SHA-1 の両方のヘッダを指定することを意味している。
名前：Public-Key-Pinning

[3] HSTS と Public Key Pinning を同時に設定する場合には、一つの setenv.add-response-header 内に両方の設定を追加すること

値　：　pin-sha256="証明書の公開鍵情報の SHA-256 ハッシュ値（pinned　fingerprint）の Base64 値", pin-sha256="バックアップのための公開鍵情報の SHA-256 ハッシュ値（backup pinned fingerprint）の Base64 値",max-age=有効期間

6）「OK」をクリックする。