Secure Socket Layer HTTP の設定

- Secure Socket Layer HTTP について （1 ページ）
- Secure Socket Layer HTTP の設定方法 （5 ページ）
- セキュア HTTP サーバおよびクライアントのステータスのモニタリング （12 ページ）
- Secure Socket Layer HTTP に関するその他の参考資料 （13 ページ）
- Secure Socket Layer HTTP に関する機能情報 （13 ページ）

Secure Socket Layer HTTP について

セキュア HTTP サーバおよびクライアントの概要

セキュア HTTP 接続の場合、HTTP サーバが送受信するデータは暗号化されてインターネットに送信されます。SSL 暗号化を伴う HTTP は、Web ブラウザからスイッチを設定するような機能に、セキュアな接続を提供します。シスコが実装するセキュア HTTP サーバおよび HTTP クライアントでは、アプリケーション層の暗号化に SSL バージョン 3.0 を使用します。HTTP over SSL は、HTTPS と省略されます（セキュアな接続の場合、URL が http://の代わりに https://で始まります）。

（注） SSL は 1999 年に Transport Layer Security（TLS）に発展しましたが、このような特定のコンテキストでまだ使用されています。

セキュア HTTP サーバ（スイッチ）の主な役割は、指定のポート（デフォルトの HTTPS ポートは 443）で HTTPS 要求を待ち受けて、HTTP 1.1 Web サーバへその要求を渡すことです。HTTP 1.1 サーバはその要求を処理して、セキュア HTTP サーバへ応答（呼び出す）します。セキュア HTTP サーバは HTTP 1.1 サーバの代わりに、元の要求に応えます。

セキュア HTTP クライアント（Web ブラウザ）の主な役割は、Cisco IOS アプリケーション要求に応答して、そのアプリケーションが要求した HTTPS User Agent サービスを実行し、応答を（そのアプリケーションに）返すことです。

CA のトラストポイント

認証局（CA）は、要求を認可して参加するネットワーク デバイスに証明書を発行します。これらのサービスは、参加するデバイスに対する中央集中的なセキュリティ キーおよび証明書の管理を提供します。特定の CA サーバはトラストポイントと呼ばれます。

接続が実行されると、HTTPS サーバは、トラストポイントとなる特定の CA から得た X.509v3 の証明書を発行することで、セキュアな接続をクライアントに提供します。クライアント（通常、Web ブラウザ）は、その証明書の認証に必要な公開キーを保有しています。

セキュア HTTP 接続では、CA トラストポイントを設定することを強く推奨します。HTTPS サーバを実行しているデバイスに CA のトラストポイントが設定されていない場合、サーバは自身を認証して必要な RSA のキーのペアを生成します。自身で認証した（自己署名）証明書は適切なセキュリティではないので、接続するクライアントはその証明書が自己証明書であることを通知し、ユーザに接続の選択（確立または拒否）をさせる必要があります。この選択肢は内部ネットワーク トポロジ（テスト用など）に役立ちます。

CA のトラストポイントを設定していない場合、セキュア HTTP 接続を有効にすると、そのセキュア HTTP サーバ（またはクライアント）に対する一時的または永続的な自己署名証明書が自動的に生成されます。

- スイッチにホスト名とドメイン名が設定されてない場合、生成される自己署名証明書は一時的なものです。スイッチがリブートすると、一時的な自己署名証明書がすべて失われて、新しい一時的な自己署名証明書が割り当てられます。
- スイッチにホスト名とドメイン名が設定されている場合、生成される自己署名証明書は永続的なものです。この証明書は、スイッチを再起動しても、セキュア HTTP サーバを無効にしても有効のままです。そのため、再度セキュア HTTP 接続を有効にしたときに使用できます。

（注） 認証局およびトラストポイントは、個々のデバイスで設定する必要があります。他のデバイスからコピーすると、それらはスイッチ上で無効になります。

新しい証明書を登録した場合、新しい設定の変更は、サーバが再起動するまで HTTPS サーバに適用されません。CLI を使用するか、または物理的な再起動によって、サーバを再起動できます。サーバを再起動すると、スイッチは新しい証明書の使用を開始します。

自己署名証明書が生成された場合、その情報は **show running-config** 特権 EXEC コマンドで出力できます。自己署名証明書を表示するコマンドの出力（show running-config コマンド）を例として一部示します。

```
Device# show running-config
Building configuration...

<output truncated>

crypto pki trustpoint TP-self-signed-3080755072
 enrollment selfsigned
 subject-name cn=IOS-Self-Signed-Certificate-3080755072
```

```
 revocation-check none
 rsakeypair TP-self-signed-3080755072
!
!
crypto ca certificate chain TP-self-signed-3080755072
 certificate self-signed 01
  3082029F 30820208 A0030201 02020101 300D0609 2A864886 F70D0101 04050030
  59312F30 2D060355 04031326 494F532D 53656C66 2D536967 6E65642D 43657274
  69666963 6174652D 33303830 37353530 37323126 30240609 2A864886 F70D0109
  02161743 45322D33 3535302D 31332E73 756D6D30 342D3335 3530301E 170D3933
  30333031 30303030 35395A17 0D323030 31303130 30303030 305A3059 312F302D

<output truncated>
```

自己署名証明書は、セキュア HTTP サーバを無効にして、**no crypto pki trustpoint TP-self-signed-30890755072** グローバル コンフィギュレーション コマンドを入力することで削除できます。その後、セキュア HTTP サーバを再度有効にすると、自己署名証明書が新たに生成されます。

（注） *TP self-signed* の後ろに表示されている値は、デバイスのシリアル番号によって異なります。

オプションのコマンド（**ip http secure-client-auth**）を使用すると、HTTPS サーバがクライアントからの X.509v3 証明書を要求します。クライアントの認証は、サーバ自身の認証よりも高いセキュリティを提供します。

認証局の詳細については、『*Cisco IOS Security Configuration Guide, Release 12.4*』の「Configuring Certification Authority Interoperability」の章を参照してください。

CipherSuite

CipherSuite は暗号化アルゴリズムおよびダイジェストアルゴリズムを指定して、SSL 接続に使用します。HTTPS サーバに接続すると、クライアントの Web ブラウザは、サポート対象の CipherSuite のリストを提供します。その後クライアントとサーバは、両方でサポートされている暗号化アルゴリズムで最適なものをリストから選択してネゴシエートします。たとえば、Netscape Communicator 4.76 は、米国のセキュリティ（RSA 公開キー暗号 MD2、MD5、RC2-CBC、RC4、DES-CBC、および DES-EDE3-CBC）をサポートしています。

最適な暗号化には、128 ビット暗号化をサポートするクライアントブラウザ（Microsoft Internet Explorer バージョン 5.5 以降または Netscape Communicator バージョン 4.76 以降など）が必要です。SSL_RSA_WITH_DES_CBC_SHA CipherSuite は、128 ビット暗号化を提供しないため、他の CipherSuite よりもセキュリティが低くなります。

CipherSuite は、よりセキュリティが高く、複雑になればなるほど、わずかですが処理時間が必要になります。次に、スイッチでサポートされる CipherSuite およびルータの処理負荷（速さ）による CipherSuite のランク（速い順）を定義します。

1. SSL_RSA_WITH_DES_CBC_SHA：メッセージの暗号化に DES-CBC、およびメッセージダイジェストに SHA を使用した RSA のキー交換（RSA 公開キー暗号化）
2. SSL_RSA_WITH_NULL_SHA：メッセージの暗号化に NULL、およびメッセージ ダイジェストに SHA を使用したキー交換（SSL 3.0 専用）。

3. SSL_RSA_WITH_NULL_MD5：メッセージの暗号化に NULL、およびメッセージ ダイジェストに MD5 を使用したキー交換（SSL 3.0 専用）。
4. SSL_RSA_WITH_RC4_128_MD5：RC4 128 ビット暗号化、およびメッセージ ダイジェストに MD5 を使用した RSA のキー交換
5. SSL_RSA_WITH_RC4_128_SHA：RC4 128 ビット暗号化、およびメッセージ ダイジェストに SHA を使用した RSA のキー交換
6. SSL_RSA_WITH_3DES_EDE_CBC_SHA：メッセージの暗号化に 3DES と DES-EDE3-CBC、およびメッセージ ダイジェストに SHA を使用した RSA のキー交換（RSA 公開キー暗号化）
7. SSL_RSA_WITH_AES_128_CBC_SHA：AES 128 ビット暗号化、およびメッセージダイジェストに SHA を使用した RSA のキー交換（SSL 3.0 専用）。
8. SSL_RSA_WITH_AES_256_CBC_SHA：AES 256 ビット暗号化、およびメッセージダイジェストに SHA を使用した RSA のキー交換（SSL 3.0 専用）。
9. SSL_RSA_WITH_AES_128_CBC_SHA：AES 128 ビット暗号化、およびメッセージダイジェストに SHA を使用した RSA のキー交換（SSL 3.0 専用）。
10. SSL_RSA_WITH_AES_256_CBC_SHA：AES 256 ビット暗号化、およびメッセージダイジェストに SHA を使用した RSA のキー交換（SSL 3.0 専用）。

（注） Chrome の最新バージョンは 4 つの元の暗号スイートをサポートしません。そのため、Web GUI とゲスト ポータル両方へのアクセスが拒否されます。

（暗号化およびダイジェスト アルゴリズムをそれぞれ指定して組み合わせた）RSA は、SSL 接続においてキーの生成および認証の両方に使用されます。これは、CA のトラストポイントが設定されているかどうかにかかわりません。

SSL のデフォルト設定

標準の HTTP サーバはイネーブルに設定されています。

SSL はイネーブルに設定されています。

CA のトラストポイントは設定されていません。

自己署名証明書は生成されていません。

SSL の設定時の注意事項

SSL をスイッチ クラスタで使用すると、SSL セッションがクラスタ コマンダで終了します。クラスタ メンバのスイッチは標準の HTTP で動作させる必要があります。

CA のトラストポイントを設定する前に、システム クロックが設定されていることを確認してください。クロックが設定されていないと、不正な日付により証明書が拒否されます。

スイッチ スタック内のスタック マスターで、SSL セッションが強制終了されます。

Secure Socket Layer HTTP の設定方法

CA のトラストポイントの設定

セキュア HTTP 接続には、CA のトラストポイントを正式に設定することを推奨します。CA のトラストポイントは、自己署名証明書より高いセキュリティがあります。

CA のトラストポイントを設定するには、特権 EXEC モードで次の手順を実行します。

手順

	コマンドまたはアクション	目的
ステップ 1	**configureterminal** 例： `Device# configure terminal`	グローバル コンフィギュレーション モードを開始します。
ステップ 2	**hostname** *hostname* 例： `Device(config)# hostname your_hostname`	スイッチのホスト名を指定します（以前ホスト名を設定していない場合のみ必須）。ホスト名はセキュリティ キーと証明書に必要です。
ステップ 3	**ip domain-name** *domain-name* 例： `Device(config)# ip domain-name your_domain`	スイッチの IP ドメイン名を指定します（以前 IP ドメイン名を設定していない場合のみ必須）。IP ドメイン名はセキュリティ キーと証明書に必要です。
ステップ 4	**crypto key generate rsa** 例： `Device(config)# crypto key generate rsa`	（任意）RSA キーペアを生成します。RSA キーのペアは、スイッチの証明書を入手する前に必要です。RSA キーのペアは自動的に生成されます。必要であれば、このコマンドを使用してキーを再生成できます。
ステップ 5	**crypto ca trustpoint** 名前 例：	CA のトラストポイントにローカルの設定名を指定して、CA トラストポイ

	コマンドまたはアクション	目的
	`Device(config)# crypto ca trustpoint your_trustpoint`	ントコンフィギュレーションモードを開始します。
ステップ 6	**enrollment url** *URL* 例： `Device(ca-trustpoint)# enrollment url http://your_server:80`	スイッチによる証明書要求の送信先の URL を指定します。
ステップ 7	**enrollment http-proxy** *host-name port-number* 例： `Device(ca-trustpoint)# enrollment http-proxy your_host 49`	（任意）HTTP プロキシ サーバを経由して CA から証明書を入手するようにスイッチを設定します。 • *host-name* には、CA を取得するために使用するプロキシサーバを指定します。 • *port-number* には、CA にアクセスするために使用するポート番号を指定します。
ステップ 8	**crlquery** *URL* 例： `Device(ca-trustpoint)# crl query ldap://your_host:49`	ピアの証明書が取り消されていないかを確認するために、証明書失効リスト（CRL）を要求するようにスイッチを設定します。
ステップ 9	**primary** 名前 例： `Device(ca-trustpoint)# primary your_trustpoint`	（任意）トラストポイントが CA 要求に対してプライマリ（デフォルト）トラストポイントとして使用されるように指定します。 • *name* には、設定したトラストポイントを指定します。
ステップ 10	**exit** 例： `Device(ca-trustpoint)# exit`	CA トラストポイントコンフィギュレーションモードを終了し、グローバルコンフィギュレーションモードに戻ります。
ステップ 11	**crypto ca authentication** 名前 例： `Device(config)# crypto ca authentication your_trustpoint`	CA の公開キーを取得して CA を認証します。ステップ 5 で使用した名前と同じものを使用します。

	コマンドまたはアクション	目的
ステップ 12	**crypto ca enroll** 名前 例： `Device(config)# crypto ca enroll your_trustpoint`	指定した CA トラストポイントから証明書を取得します。このコマンドは、各 RSA キーのペアに対して 1 つの署名入りの証明書を要求します。
ステップ 13	**end** 例： `Device(config)# end`	特権 EXEC モードに戻ります。

セキュア HTTP サーバの設定

セキュア HTTP サーバを設定するには、特権 EXEC モードで次の手順を実行します。

始める前に

証明に証明書の認証を使用する場合、前の手順を使用してスイッチの CA トラストポイントを設定してから、HTTP サーバを有効にする必要があります。CA のトラストポイントを設定していない場合、セキュア HTTP サーバを最初に有効にした時点で、自己署名証明書が生成されます。サーバを設定した後、標準およびセキュア HTTP サーバ両方に適用するオプション（パス、適用するアクセス リスト、最大接続数、またはタイムアウト ポリシー）を設定できます。

Web ブラウザを使用してセキュア HTTP 接続を確認するには、https://*URL* を入力します（URL は IP アドレス、またはサーバ スイッチのホスト名）。デフォルト ポート以外のポートを設定している場合、URL の後ろにポート番号も指定する必要があります。次に例を示します。

（注） AES256_SHA2 はサポートされません。

```
https://209.165.129:1026
```

または

```
https://host.domain.com:1026
```

アクセス リスト（IPv4 ACL のみ）を指定するための従来の **ip http access-class** *access-list-number* コマンドは廃止予定です。引き続きこのコマンドを使用して、HTTP サーバへのアクセスを許可するアクセス リストを指定できます。2 つの新しいコマンドは、IPv4 および IPv6 ACL を指定するためのサポートを有効にするために導入されました。これらは、IPv4 ACL を指定するための **ip http access-class ipv4** *access-list-name* | *access-list-number* と、IPv6 ACL を指定する

ための **ip http access-class ipv6** *access-list-name* です。警告メッセージの受信を防ぐために、新しい CLI の使用をお勧めします。

アクセス リストを指定する際は、次の考慮事項があります。

- 存在しないアクセス リストを指定すると、設定は実行されますが、次の警告メッセージを受信します。

  ```
  ACL being attached does not exist, please configure it
  ```

- HTTP サーバにアクセス リストを指定するために **ip http access-class** コマンドを使用すると、次の警告メッセージが表示されます。

  ```
  This CLI will be deprecated soon, Please use new CLI ip http
  access-class ipv4/ipv6 <access-list-name>| <access-list-number>
  ```

- **ip http access-class ipv4** *access-list-name* | *access-list-number* または **ip http access-class ipv6** *access-list-name* を使用する場合、**ip http access-class** を使用してアクセス リストがすでに設定されていると、次の警告メッセージが表示されます。

  ```
  Removing ip http access-class <access-list-number>
  ```

ip http access-class *access-list-number* と **ip http access-class ipv4** *access-list-name* | *access-list-number* は同じ機能を共有します。コマンドを実行するごとに、その前のコマンドのコンフィギュレーションは上書きされます。2 つのコマンドの設定間の次の組み合わせによって、実行コンフィギュレーションへの影響が説明されます。

- **ip http access-class** *access-list-number* がすでに設定されている場合、**ip http access-class ipv4** *access-list-number* コマンドを使用して設定しようとすると、**ip http access-class** *access-list-number* の設定が削除され、**ip http access-class ipv4** *access-list-number* の設定が実行コンフィギュレーションに追加されます。

- **ip http access-class** *access-list-number* がすでに設定されている場合、**ip http access-class ipv4** *access-list-name* コマンドを使用して設定しようとすると、**ip http access-class** *access-list-number* の設定が削除され、**ip http access-class ipv4** *access-list-name* の設定が実行コンフィギュレーションに追加されます。

- **ip http access-class ipv4** *access-list-number* がすでに設定されている場合、**ip http access-class** *access-list-name* を使用して設定しようとすると、**ip http access-class ipv4** *access-list-number* の設定がコンフィギュレーションから削除され、**ip http access-class** *access-list-name* の設定が実行コンフィギュレーションに追加されます。

- **ip http access-class ipv4** *access-list-name* がすでに設定されている場合、**ip http access-class** *access-list-number* を使用して設定しようとすると、**ip http access-class ipv4** *access-list-name* の設定がコンフィギュレーションから削除され、**ip http access-class** *access-list-number* の設定が実行コンフィギュレーションに追加されます。

手順

	コマンドまたはアクション	目的

ステップ 1	**show ip http server status**
	例：
	`Device# show ip http server status`
		（任意）HTTP サーバのステータスを表示して、セキュア HTTP サーバの機能がソフトウェアでサポートされているかどうかを判断します。出力で、次のラインのどちらかを確認してください。
		`HTTP secure server capability: Present`
		または
		`HTTP secure server capability: Not present`

ステップ 2	**configure terminal** 例： `Device# configure terminal`	グローバル コンフィギュレーション モードを開始します。
ステップ 3	**ip http secure-server** 例： `Device(config)# ip http secure-server`	HTTPS サーバがディセーブルの場合、イネーブルにします。HTTPS サーバは、デフォルトでイネーブルに設定されています。
ステップ 4	**ip http secure-port** *port-number* 例： `Device(config)# ip http secure-port 443`	（任意）HTTPS サーバに使用するポート番号を指定します。デフォルトのポート番号は 443 です。443 または 1025 ～ 65535 の範囲で指定できます。
ステップ 5	**ip http secure-ciphersuite** {[**3des-ede-cbc-sha**] [**rc4-128-md5**] [**rc4-128-sha**] [**des-cbc-sha**]} 例： `Device(config)# ip http secure-ciphersuite rc4-128-md5`	（任意）HTTPS 接続の暗号化に使用する CipherSuite（暗号化アルゴリズム）を指定します。特定の CipherSuite を指定する理由がなければ、サーバとクライアントが、両方がサポートする CipherSuite でネゴシエートするように設定します。これがデフォルトです。

	コマンドまたはアクション	目的
ステップ 6	**ip http secure-client-auth** 例： `Device(config)# ip http secure-client-auth`	（任意）HTTP サーバを設定して、接続処理の間、認証のために、クライアントからの X.509v3 証明書を要求します。デフォルトでは、クライアントがサーバからの証明書を要求する設定になっていますが、サーバはクライアントを認証しないようになっています。
ステップ 7	**ip http secure-trustpoint** 名前 例： `Device(config)# ip http secure-trustpoint your_trustpoint`	X.509v3 セキュリティ証明書の取得およびクライアントの証明書接続の認証に使用する CA のトラストポイントを指定します。 （注） このコマンドの使用は、前の手順に従って CA のトラストポイントをすでに設定しているという前提を踏まえて説明しています。
ステップ 8	**ip http path** *path-name* 例： `Device(config)# ip http path /your_server:80`	（任意）HTML ファイルのベースとなる HTTP パスを設定します。パスは、ローカル システムにある HTTP サーバファイルの場所を指定します（通常、システムのフラッシュメモリを指定します）。
ステップ 9	**ip http access-class** *access-list-number* 例： `Device(config)# ip http access-class 2`	（任意）HTTP サーバへのアクセスの許可に使用するアクセスリストを指定します。
ステップ 10	**ip http access-class** { **ipv4** {*access-list-number* | *access-list-name*} | **ipv6** {*access-list-name*} } 例： `Device(config)# ip http access-class ipv4 4`	（任意）HTTP サーバへのアクセスの許可に使用するアクセスリストを指定します。
ステップ 11	**ip http max-connections** *value* 例： `Device(config)# ip http max-connections 4`	（任意）HTTP サーバへの同時最大接続数を指定します。値は10以上にすることを推奨します。これは、UI が想定どおりに機能するために必要な値です。

	コマンドまたはアクション	目的
ステップ 12	**ip http timeout-policyidle** *seconds***life** *seconds***requests** *value* 例： `Device(config)# ip http timeout-policy idle 120 life 240 requests 1`	（任意）指定の状況下における、HTTP サーバへの接続最大時間を指定します。 • **idle**：データの受信がないか、応答データが送信できない場合の最大時間。範囲は 1 ～ 600 秒です。デフォルト値は180秒（3分）です。 • **life**：接続を確立している最大時間。指定できる範囲は 1 ～ 86400 秒（24時間）です。デフォルトは 180 秒です。 • **requests**：永続的な接続で処理される要求の最大数。最大値は 86400 です。デフォルトは 1 です。
ステップ 13	**end** 例： `Device(config)# end`	特権 EXEC モードに戻ります。

セキュア HTTP クライアントの設定

セキュア HTTP クライアントを設定するには、特権 EXEC モードで次の手順を実行します。

始める前に

標準の HTTP クライアントおよびセキュア HTTP クライアントは常にイネーブルです。証明書の認証にはセキュア HTTP クライアントの証明書が必要です。次の手順では、前の手順で CA のトラストポイントをスイッチに設定していることを前提にしています。CA のトラストポイントが設定されておらず、リモートの HTTPS サーバがクライアントの認証を要求した場合、セキュア HTTP クライアントへの接続は失敗します。

手順

	コマンドまたはアクション	目的
ステップ 1	**configureterminal** 例： `Device# configure terminal`	グローバル コンフィギュレーション モードを開始します。

	コマンドまたはアクション	目的
ステップ 2	**ip http client secure-trustpoint** 名前 例： `Device(config)# ip http client secure-trustpoint your_trustpoint`	（任意）リモートの HTTP サーバがクライアント認証を要求した場合に使用する、CA のトラストポイントを指定します。このコマンドの使用は、前の手順を使用して CA のトラストポイントをすでに設定しているという前提を踏まえて説明しています。クライアント認証が必要ない場合、またはプライマリのトラストポイントがすでに設定されている場合は、このコマンドは任意です。
ステップ 3	**ip http client secure-ciphersuite** {[**3des-ede-cbc-sha**] [**rc4-128-md5**] [**rc4-128-sha**] [**des-cbc-sha**]} 例： `Device(config)# ip http client secure-ciphersuite rc4-128-md5`	（任意）HTTPS 接続の暗号化に使用する CipherSuite（暗号化アルゴリズム）を指定します。特定の CipherSuite を指定する理由がなければ、サーバとクライアントが、両方がサポートする CipherSuite でネゴシエートするように設定します。これがデフォルトです。
ステップ 4	**end** 例： `Device(config)# end`	特権 EXEC モードに戻ります。

セキュア HTTP サーバおよびクライアントのステータスのモニタリング

SSL セキュア サーバおよびクライアントのステータスをモニタするには、次の表の特権 EXEC コマンドを使用します。

表 *1 : SSL* セキュア サーバおよびクライアントのステータスを表示するコマンド

コマンド	目的
show ip http client secure status	セキュア HTTP クライアントの設定を表示します。
show ip http server secure status	セキュア HTTP サーバの設定を表示します。
show running-config	セキュア HTTP 接続に対して生成された自己署名証明書を表示します。

Secure Socket Layer HTTP に関するその他の参考資料

関連資料

関連項目	参照先
証明機関	「Configuring Certification Authority Interoperability」

MIB

MIB	MIB リンク
本リリースでサポートするすべての MIB	選択したプラットフォーム、Cisco IOS リリース、およびフィーチャ セットに関する MIB を探してダウンロードするには、次の URL にある Cisco MIB Locator を使用します。 http://www.cisco.com/go/mibs

テクニカル サポート

説明	リンク
シスコのサポート Web サイトでは、シスコの製品やテクノロジーに関するトラブルシューティングにお役立ていただけるように、マニュアルやツールをはじめとする豊富なオンライン リソースを提供しています。 お使いの製品のセキュリティ情報や技術情報を入手するために、Product Alert Tool（Field Notice からアクセス）、Cisco Technical Services Newsletter、Really Simple Syndication（RSS）フィードなどの各種サービスに加入できます。 シスコのサポート Web サイトのツールにアクセスする際は、Cisco.com のユーザ ID およびパスワードが必要です。	http://www.cisco.com/support

Secure Socket Layer HTTP に関する機能情報

次の表に、このモジュールで説明した機能に関するリリース情報を示します。この表は、ソフトウェア リリース トレインで各機能のサポートが導入されたときのソフトウェア リリースのみを示しています。その機能は、特に断りがない限り、それ以降の一連のソフトウェア リリースでもサポートされます。

プラットフォームのサポートおよびCisco ソフトウェア イメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

表 *2 : Secure Socket Layer HTTP* に関する機能情報

機能名	リリース	機能情報
Secure Socket Layer HTTP	Cisco IOS XE Everest 16.5.1a	シスコが実装するセキュア HTTP サーバおよび HTTP クライアントでは、アプリケーション層の暗号化に SSL バージョン 3.0 を使用します。セキュア HTTP 接続の場合、HTTP サーバが送受信するデータは暗号化されてインターネットに送信されます。 この機能は、次のプラットフォームに実装されていました。 • Cisco Catalyst 9500 シリーズ スイッチ