CSS11500 でのグローバル証明書の要求とインストール

目次

概要

コンテンツ サービス スイッチ（CSS）の既存のキーと証明書がない場合、それらを CSS で生成できます。 CSS には、秘密キー、証明書署名要求（CSR）、および自己署名仮証明書を生成するプロセスを簡略化するための一連の証明書と秘密キーの管理ユーティリティが含まれています。 このドキュメントでは、認証局（CA）から新しい証明書を取得して、CSS にインストールする手順について説明します。

前提条件

要件

このドキュメントに関する固有の要件はありません。

使用するコンポーネント

このドキュメントは、特定のソフトウェアやハードウェアのバージョンに限定されるものではありません。

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されたものです。 このドキュメントで使用するすべてのデバイスは、クリアな（デフォルト）設定で作業を開始しています。 ネットワークが稼働中の場合は、コマンドが及ぼす潜在的な影響を十分に理解しておく必要があります。

表記法

ドキュメント表記の詳細は、『シスコ テクニカル ティップスの表記法』を参照してください。

設定

この項では、このドキュメントで説明する機能の設定に必要な情報を提供します。

注: このドキュメントで使用されているコマンドの詳細を調べるには、Command Lookup Tool（登録ユーザ専用）を使用してください。

設定

このドキュメントでは、次の設定を使用します。

- Rivest、Shamir、Adelman（RSA）キー ペアの生成
- RSA キー ペア ファイルの関連付け
- CSR の生成
- Verisign 中間証明書の取得
- チェーン証明書ファイルのインポート
- 証明書ファイルの関連付け
- SSL プロキシ リストの設定
- セキュア ソケット レイヤ（SSL）サービスとコンテンツ ルールの設定

Rivest、Shamir、Adelman（RSA）キー ペアの生成

ssl genrsa コマンドを発行して、非対称暗号化用に RSA 秘密鍵/公開鍵キー ペアを生成します。CSS は CSS でファイルとして生成された RSA キー ペアを保存します。 たとえば、RSA キー ペア myrsakey.pem を生成するには、次のように入力します。

```
CSS11500(config) # ssl genrsa myrsakey.pem 1024 "passwd123" Please be patient this could take a few minutes
```

RSA のキー ペア ファイルの関連付け

ssl associate rsakey コマンドを発行し、生成された RSA キー ペアに RSA キー ペア名を関連付けます。 たとえば、生成された RSA キー ペア ファイル myrsakey.pem に RSA キー名 myrsakey1 を関連付けるには、次のように入力します。

```
CSS11500(config) # ssl associate rsakey myrsakey1 myrsakey.pem
```

CSR の生成

ssl gencsr rsakey コマンドを発行し、関連付けをした RSA キー ペア ファイルの CSR ファイルを生成します。 この CSR は、署名用に CA に送信されます。 たとえば、RSA キー ペア myrsakey1 に基づいて CSR を生成するには、次のように入力します。

```
CSS11503(config)# ssl gencsr myrsakey1 You are about to be asked to enter information that will be incorporated into your certificate request. What you are about to enter is what is called a Distinguished Name or a DN. For some fields there will be a default value, If you enter '.', the field will be left blank. Country Name (2 letter code) [US] US State or Province (full name)
```

```
[SomeState] CA Locality Name (city) [SomeCity] San Jose Organization Name (company name) [Acme
Inc]Cisco Systems, Inc. Organizational Unit Name (section) [Web Administration] Web Admin Common
Name (your domain name) [www.acme.com] www.cisco.com Email address [webadmin@acme.com]
webadmin@cisco.com
```

ssl gencsr コマンドにより、画面に CSR および出力が生成されます。 ほとんどの主要な CA には、画面に証明書リクエストをカット アンド ペーストすることを求める Web ベースのアプリケーションが実装されています。

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBWDCCAQICAQAwgZwxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJNQTETMBEGA1UE
BxMKQm94Ym9yb3VnaDEcMBoGA1UEChMTQ2lzY28gU3lzdGVtcywgSW5jLjESMBAG
A1UECxMJV2ViIEFkbWluMRYwFAYDVQQDEw13d3cuY2lzY28uY29tMSEwHwYJKoZI
hvcNAQkBFhJra3JvZWJlckBjaXNjby5jb20wXDANBgkqhkiG9w0BAQEFAANLADBI
AkEAqHXjtQUVXvmo6tAWPiMpe6oYhZbJUDgTxbW4VMCygzGZn2wUJTgLrifDB6N3
v+1tKFndE686BhKqfyOidml3wQIDAQABoAAwDQYJKoZIhvcNAQEEBQADQQA94yC3
4SUJJ4UQEnO2OqRGLOZpAElc4+IV9aTWK6NmiZsM9Gt0vPhIkLx5jjhVRLlb27Ak
H6D5omXa0SPJan5x
-----END CERTIFICATE REQUEST-----
```

CA によって、署名した CSR が返されます。この処理は、通常、CSR で提供される電子メール アドレスを使用して行われます。

[Verisign 中間証明書の取得]

CA からの証明書の取得

CA に CSR を送信した後、1 ~ 7 営業日の間に、署名済み証明書が届きます。 所要日数は、CA によって異なります。 CA が署名して提供した証明書は、CSS に追加できます。

ステップアップ/SGC またはチェーン証明書に適用する場合、証明書に署名する中間証明書を取得する必要があります。 次のリンクから VeriSign の中間証明書を取得できます。

- 中間 CA 証明書のインストール

中間証明書をファイルに保存します たとえば、intermediate.pem に保存します。

サーバ証明書と中間証明書の結合

CSS でチェーン証明書を使用するには、サーバ証明書と中間証明書を結合する必要があります。これにより、CSS は、最初の SSL ハンドシェイクにおいて、クライアントに完全な証明書チェーンを返すことができます。 CSS のチェーン証明書ファイルを作成する場合、証明書が正しい順序であることを確認します。 最初にサーバ証明書が配置される必要があり、サーバ証明書の署名に使用される中間証明書はその次に配置される必要があります。 サーバ証明書と中間証明書の間に、空白行を 1 行挿入します。 たとえば、サーバ証明書 servercert.pem と中間証明書 intermediate.pem を結合して、mychainedrsacert.pem というチェーン証明書を作成します。 次に、mychainedrsacert.pem の内容全体を示します。

```
-----BEGIN CERTIFICATE-----
MIICwTCCAioCAQUwDQYJKoZIhvcNAQEEBQAwgagxCzAJBgNVBAYTAlVTMRMwEQYD
 VQQIEwpDYWxpZm9ybmlhMREwDwYDVQQHEwhTYW4gSm9zZTEeMBwGA1UEChMVRXhh
 bXBsZSBTeXN0ZW1zLCBJbmMuMRIwEAYDVQQLEwlXZWIgQWRtaW4xGDAWBgNVBAMT
 D3d3dy5leGFtcGxlLmNvbTEjMCEGCSqGSIb3DQEJARYUd2ViYWRtaW5AZXhhbXBs
 ZS5jb20wHhcNMDQwMTA5MDgzMjI3WhcNMDQwMjA4MDgzMjI3WjCBqDELMAkGA1UE
 BhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExETAPBgNVBAcTCFNhbiBKb3NlMR4w
 HAYDVQQKExVFeGFtcGxlIFN5c3RlbXMsIEluYy4xEjAQBgNVBAsTCVdlYiBBZG1p
```

```
 bjEYMBYGA1UEAxMPd3d3LmV4YW1wbGUuY29tMSMwIQYJKoZIhvcNAQkBFhR3ZWJh
 ZG1pbkBleGFtcGxlLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA2huF
 xhVeODHmoXJ4HulDqVQtCvX7eERyRarNI71p0ZV+q+qGYRtJdrlzUav/TbRn5dc0
 8IXjqrASAtTo2S4eWlTOJUnR2g0LH/lcPUaF8f+m+eODWoT8dCtNA5sgEnINAR2y
 HlS5j6dZNcyMY0nFOh68oRsZJJ58u0ZPJjl6eAsCAwEAATANBgkqhkiG9w0BAQQF
 AAOBgQADO/UTIIHnIq2Q0ICiqAQju9nz1vTiIYHBpBnUd8NkPhIHIOqNn9iZ5Q+a
 2zFjh+N2uEt5NxNOEZRbrTZH+HmZMsqJJfvfd62iq+636aPIcoo7X541DYotM05C
 OQjnehsjgwziKlp6UJtuiAwwaxtMIbP7lQXHGO6E9RnzQSvQGQ==
 -----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
 MIIDgzCCAuygAwIBAgIQJUuKhThCzONY+MXdriJupDANBgkqhkiG9w0BAQUFADBf
 MQswCQYDVQQGEwJVUzEXMBUGA1UEChMOVmVyaVNpZ24sIEluYy4xNzA1BgNVBAsT
 LkNsYXNzIDMgUHVibGljIFByaW1hcnkgQ2VydGlmaWNhdGlvbiBBdXRob3JpdHkw
 HhcNOTcwNDE3MDAwMDAwWhcNMTExMDI0MjM1OTU5WjCBujEfMB0GA1UEChMWVmVy
 aVNpZ24gVHJ1c3QgTmV0d29yazEXMBUGA1UECxMOVmVyaVNpZ24sIEluYy4xMzAx
 BgNVBAsTKlZlcmlTaWduIEludGVybmF0aW9uYWwgU2VydmVyIENBIC0gQ2xhc3Mg
 MzFJMEcGA1UECxNAd3d3LnZlcmlzaWduLmNvbS9DUFMgSW5jb3JwLmJ5IFJlZi4g
 TElBQklMSVRZIExURC4oYyk5NyBWZXJpU2lnbjCBnzANBgkqhkiG9w0BAQEFAAOB
 jQAwgYkCgYEA2IKA6NYZAn0fhRg5JaJlK+G/1AXTvOY2O6rwTGxbtueqPHNFVbLx
 veqXQu2aNAoV1Klc9UAl3dkHwTKydWzEyruj/lYncUOqY/UwPpMo5frxCTvzt01O
 OfdcSVq4wR3Tsor+cDCVQsv+K1GLWjw6+SJPkLICp1OcTzTnqwSye28CAwEAAaOB
 4zCB4DAPBgNVHRMECDAGAQH/AgEAMEQGA1UdIAQ9MDswOQYLYIZIAYb4RQEHAQEw
 KjAoBggrBgEFBQcCARYcaHR0cHM6Ly93d3cudmVyaXNpZ24uY29tL0NQUzA0BgNV
 HSUELTArBggrBgEFBQcDAQYIKwYBBQUHAwIGCWCGSAGG+EIEAQYKYIZIAYb4RQEI
 ATALBgNVHQ8EBAMCAQYwEQYJYIZIAYb4QgEBBAQDAgEGMDEGA1UdHwQqMCgwJqAk
 oCKGIGh0dHA6Ly9jcmwudmVyaXNpZ24uY29tL3BjYTMuY3JsMA0GCSqGSIb3DQEB
 BQUAA4GBAAgB7ORolANC8XPxI6I63unx2sZUxCM+hurPajozq+qcBBQHNgYL+Yhv
 1RPuKSvD5HKNRO3RrCAJLeH24RkFOLA9D59/+J4C3IYChmFOJl9en5IeDCSk9dBw
 E88mw0M9SR2egi5SX7w+xmYpAY5Okiy8RnUDgqxz6dl+C2fvVFIa
 -----END CERTIFICATE-----
```

チェーン証明書ファイルのインポート

CA によって CSR への署名が行われると、「証明書」と呼ばれるものが完成します。 証明書ファイルは CSS にインポートする必要があります。 **copy ssl コマンド**を発行して、CSS からの証明書および秘密鍵のインポート、または CSS への証明書および秘密鍵のエクスポートを実行します。 CSS は、インポートされたすべてのファイルを、CSS 上の安全な場所に保存します。 このコマンドは、SuperUser モードに限り使用できます。 たとえば、リモート サーバから CSS に mychainedrsacert.pem 証明書をインポートするには、次のように入力します。

```
CSS11500# copy ssl sftp ssl_record import mychainedrsacert.pem PEM “passwd123” Connecting
Completed successfully
```

証明書ファイルの関連付け

ssl associate cert コマンドを発行して、証明書名をインポートされた証明書に関連付けます。 たとえば、証明書名 mychainedrsacert1 をインポートされた証明書ファイル mychainedrsacert.pem に関連付けるには、次のように入力します。

```
CSS11500(config)# ssl associate cert mychainedrsacert1 mychainedrsacert.pem
```

SSL プロキシ リストの設定

ssl-proxy-list コマンドを発行し、SSL プロキシ リストを変更します。 SSL プロキシ リストは、SSL サービスに関連付けられている、関連する仮想 SSL サーバまたはバックエンド SSL サーバのグループです。 SSL プロキシ リストには、各仮想 SSL サーバに関するすべての設定情報が含まれます。 この情報には、SSL サーバの作成、証明書および対応する SSL キー ペア、仮想 IP (VIP) アドレスおよびポート、サポートされている SSL 暗号化、および他の SSL オプション

が含まれます。 たとえば、ssl-proxy-list ssl_list1 を作成するには、次のように入力します。

```
CSS11500(config)# ssl-proxy-list ssl_list1 Create ssl-list <ssl_list1>, [y/n]: y
```

SSL プロキシ リストを作成すると、CLI は ssl-proxy-list コンフィギュレーション モードになります。 次に示すように、SSL サーバを設定します。

```
CSS11500(ssl-proxy-list[ssl_list1])# ssl-server 20 CSS11500(ssl-proxy-list[ssl_list1])# ssl-server 20 vip address 192.168.3.6 CSS11500(ssl-proxy-list[ssl_list1])# ssl-server 20 rsacert mychainedrsacert1 CSS11500(ssl-proxy-list[ssl_list1])# ssl-server 20 rsakey myrsakey1 CSS11500(ssl-proxy-list[ssl_list1])# ssl-server 20 cipher rsa-export-with-rc4-40-md5 192.168.11.2 80 5 CSS11500(ssl-proxy-list[ssl_list1])# active
```

セキュア ソケット レイヤ (SSL) サービスとコンテンツ ルールの設定

SSL プロキシ リストが有効になったら、サービスとコンテンツ ルールを設定して、CSS が SSL モジュールに SSL トラフィックを送信できるようにする必要があります。 次の表に、仮想 SSL サーバ用に SSL サービスを作成するための手順 (概要) を示します。SSL プロキシ リストをサービスに追加する方法や、SSL コンテンツ ルールを作成する方法も記載しています。

SSL サービスの作成

```
CSS11500(config)# service ssl_serv1Create service <ssl_serv1>, [y/n]: y CSS11500(config-service[ssl_serv1])# type ssl-accel CSS11500(config-service[ssl_serv1])# slot 2 CSS11500(config-service[ssl_serv1])# keepalive type none CSS11500(config-service[ssl_serv1])# add ssl-proxy-list ssl_list1 CSS11500(config-service[ssl_serv1])# active
```

SSL コンテンツ ルールの作成

```
CSS11500(config)# owner ssl_owner Create owner <ssl_owner>, [y/n]: y CSS11500(config-owner[ssl_owner])# content ssl_rule1 Create content <ssl_rule1>, [y/n]: y CSS11500(config-owner-content[ssl-rule1]# vip address 192.168.3.6 CSS11500(config-owner-content[ssl-rule1]# port 443 CSS11500(config-owner-content[ssl_rule1])# add service ssl_serv1 CSS11500(config-owner-content[ssl_rule1])# active
```

クリア テキストのコンテンツ ルールの作成

```
CSS11500(config-owner[ssl_owner])# content decrypted_www Create content <decrypted_www>, [y/n]: y CSS11500(config-owner-content[decrypted_www]# vip address 192.168.11.2 CSS11500(config-owner-content[decrypted_www]# port 80 CSS11500(config-owner-content[decrypted_www])# add service linux_http CSS11500(config-owner-content[decrypted_www])# add service win2k_http CSS11500(config-owner-content[decrypted_www])# active
```

この時点で、クライアント HTTPS トラフィックは 192.168.3.6:443 にある CSS に送信できます。 CSS は、HTTPS トラフィックを復号化し、HTTP に変換します。 CSS ではサービスを選択し、HTTP Web サーバに HTTP トラフィックを送信します。 次に、上記の例を使用して動作している CSS の設定を示します。

```
CSS11501# show run configure !************************** GLOBAL ************************** ssl associate rsakey myrsakey1 myrsakey.pem ssl associate cert mychainedrsacert1 mychainedrsacert.pem ip route 0.0.0.0 0.0.0.0 192.168.3.1 1 ftp-record conf 192.168.11.101 admin des-password 4f2bxansrcehjgka /tftpboot !************************ INTERFACE ************************ interface 1/1 bridge vlan 10 description "Client Side" interface 1/2 bridge vlan 20 description "Server Side" !************************* CIRCUIT ************************* circuit VLAN10 description "Client Segment" ip address 192.168.3.254 255.255.255.0 circuit VLAN20 description "Server Segment" ip address 192.168.11.1 255.255.255.0 !********************** SSL PROXY LIST ********************** ssl-proxy-list ssl_list1 ssl-server 20 ssl-server 20 vip address 192.168.3.6 ssl-server 20 rsakey myrsakey1 ssl-server 20
```

```
rsacert mycertcert1 ssl-server 20 cipher rsa-with-rc4-128-md5 192.168.11.2 80 active
!************************* SERVICE ************************* service linux-http ip address
192.168.11.101 port 80 active service win2k-http ip address 192.168.11.102 port 80 active
service ssl_serv1 type ssl-accel slot 2 keepalive type none add ssl-proxy-list ssl_list1 active
!************************* OWNER ************************* owner ssl_owner content ssl_rule1
vip address 192.168.3.6 protocol tcp port 443 add service ssl_serv1 active content decrypted_www
vip address 192.168.11.2 add service linux-http add service win2k-http protocol tcp port 80
active
```

確認

ここでは、設定が正常に動作していることを確認します。

show ssl file および **show ssl associate コマンド**を使用して、設定を確認します。

すべてのファイルのサイズが 0 よりも大きいことを確認します。

clear ssl file コマンドを使用して、証明書または鍵を削除できます。

トラブルシューティング

ここでは、設定に関するトラブルシューティングについて説明します。

SSL ネゴシエーションが失敗した場合、**show ssl statistics コマンド**を使用して、失敗した SSL ネゴシエーションに関する有益な情報を表示します。

たとえば、次のフィールドをチェックします。

```
0 Unknown issuer certificates
0 Failed signatures decryptions
0 Invalid issuer keys
0 Not yet valid certificates
0 Expired Client certificates
0 Revoked certificates
0 CRLs not obtained from host
0 CRLs with bad HTTP return codes
0 CRLs not loaded because of low memory
0 CRLs obtained but failed to load
0 CRLs with invalid signatures
0 CRLs successfully loaded
0 Successful server authentications
0 Server authentications failed
0 Expired Server certificates
```

関連情報

- CSS 11500
- CSS 11000 シリーズ コンテント サービス スイッチのハードウェア サポート (英語)
- Cisco WebNS CSS11500 ソフトウェアのダウンロード (登録ユーザ専用)
- Cisco WebNS CSS11000 ソフトウェアのダウンロード (登録ユーザ専用)
- テクニカルサポートとドキュメント - Cisco Systems