Provisioning Examples

Provisioning Examples Overview

This chapter provides example procedures for transferring configuration profiles between the phone and the provisioning server.

For information about creating configuration profiles, refer to Provisioning Formats.

Basic Resync

This section demonstrates the basic resync functionality of the phones.

TFTP Resync

The phone supports multiple network protocols for retrieving configuration profiles. The most basic profile transfer protocol is TFTP (RFC1350). TFTP is widely used for the provisioning of network devices within private LAN networks. Although not recommended for the deployment of remote endpoints across the Internet, TFTP can be convenient for deployment within small organizations, for in-house preprovisioning, and for development and testing. See In-House Device Preprovisioning for more information on in-house preprovisioning. In the following procedure, a profile is modified after downloading a file from a TFTP server.

Procedure

Step 1 Within a LAN environment, connect a PC and a phone to a hub, switch, or small router.

Step 2 Connect an analog phone to the Phone 1 port of the ATA

Step 3 On the PC, install and activate a TFTP server.

Step 4 Use a text editor to create a configuration profile that sets the value for GPP_A to 12345678 as shown in the example.

```
<flat-profile>
  <GPP_A> 12345678
  </GPP_A>
</flat-profile>
```

Step 5 Save the profile with the name `basic.txt` in the root directory of the TFTP server.

You can verify that the TFTP server is properly configured: request the `basic.txt` file by using a TFTP client other than the phone. Preferably, use a TFTP client that is running on a separate host from the provisioning server.

Step 6 Using an analog phone, obtain the IP address of the ATA (IVR menu ****** 110 #**).

If the configuration has been modified since it was manufactured, perform a factory reset on the phone by using the IVR RESET option (****** 73738#**).

Step 7 Open the PC web browser. For example, if the IP address of the device is 192.168.1.100:

```
http://192.168.1.100
```

Step 8 Select the **Voice** > **Provisioning** tab, and inspect the values of the general purpose parameters GPP_A through GPP_P. These should be empty.

Step 9 Resync the test phone to the `basic.txt` configuration profile by opening the resync URL in a web browser window.

If the IP address of the TFTP server is 192.168.1.200, the command should be similar to the following example:

```
http://192.168.1.100/admin/resync?tftp://192.168.1.200/basic.txt
```

When the phone receives this command, the device at address 192.168.1.100 requests the file `basic.txt` from the TFTP server at IP address 192.168.1.200. The phone then parses the downloaded file and updates the GPP_A parameter with the value 12345678.

Step 10 Verify that the parameter was correctly updated: Refresh the configuration page on the PC web browser and select the **Voice** > **Provisioning** tab.

The GPP_A parameter should now contain the value 12345678.

Use Syslog to Log Messages

A phone can be configured to send logging messages to a syslog server over UDP, including messages related to provisioning. This server is identified in the web server administration (**Admin Login** > **Administration** > **Log** > **Debug Log Settings**, IPv4 Address field). Configure the syslog server IP address into the device and observe the messages that are generated during the remaining procedures.

To get the information, you can access the phone Web interface, select **Info** > **Debug Info** > **Control Logs** and click **messages**.

Before you begin

Procedure

Step 1 Install and activate a syslog server on the local PC.

Step 2 Program the PC IP address into the Syslog_Server_IP parameter of the profile and submit the change:

```
<Syslog_Server_IP>192.168.1.210</Syslog_Server_IP>
```

Step 3 Click the **System** tab and enter the value of your local syslog server into the Syslog_Server parameter.

Step 4 Repeat the resync operation as described in TFTP Resync, on page 1.

The device generates two syslog messages during the resync. The first message indicates that a request is in progress. The second message marks success or failure of the resync.

Step 5 Verify that your syslog server received messages similar to the following:

```
ATA192-MPP 00:0e:08:ab:cd:ef -- Successful resync tftp://192.168.1.200/basic.txt
```

The contents of these messages can be configured by using the following parameters:

- Log_Resync_Request_Msg
- Log_Resync_Success_Msg
- Log_Resync_Failure_Msg

If any of these parameters are cleared, the corresponding syslog message is not generated.

Resync a Device Automatically

A device can resync periodically to the provisioning server to ensure that any profile changes made on the server are propagated to the endpoint device (as opposed to sending an explicit resync request to the endpoint).

To cause the phone to periodically resync to a server, a configuration profile URL is defined by using the Profile_Rule parameter, and a resync period is defined by using the Resync_Periodic parameter.

Before you begin

Access the phone administration web page. See Access the Phone Web Interface.

Procedure

Step 1 Select **Voice** > **Provisioning**.

Step 2 Define the Profile_Rule parameter. This example assumes a TFTP server IP address of 192.168.1.200.

Step 3 In the **Resync Periodic** field, enter a small value for testing, such as **30** seconds.

Step 4 Click **Submit all Changes**.

With the new parameter settings, the phone resyncs twice a minute to the configuration file that the URL specifies.

Step 5 Observe the resulting messages in the syslog trace (as described in the Use Syslog to Log Messages, on page 2 section).

Step 6 Ensure that the **Resync On Reset** field is set to **Yes**.

```
<Resync_On_Reset>Yes</Resync_On_Reset>
```

Step 7 Power cycle the phone to force it to resync to the provisioning server.

If the resync operation fails for any reason, such as if the server is not responding, the unit waits (for the number of seconds configured in **Resync Error Retry Delay**) before it attempts to resync again. If **Resync Error Retry Delay** is zero, the phone does not try to resync after a failed resync attempt.

Step 8 (Optional) Set the value of **Resync Error Retry Delay** field to a small number, such as **30**.

```
<Resync_Error_Retry_Delay>30</Resync_Error_Retry_Delay>
```

Step 9 Disable the TFTP server, and observe the results in the syslog output.

Unique Profiles, Macro Expansion, and HTTP

In a deployment where each phone must be configured with distinct values for some parameters, such as User_ID or Display_Name, the service provider can create a unique profile for each deployed device and host those profiles on a provisioning server. Each phone, in turn, must be configured to resync to its own profile according to a predetermined profile naming convention.

The profile URL syntax can include identifying information that is specific to each phone, such as MAC address or serial number, by using the macro expansion of built-in variables. Macro expansion eliminates the need to specify these values in multiple locations within each profile.

A profile rule undergoes macro expansion before the rule is applied to the phone. The macro expansion controls a number of values, for example:

- $MA expands to the unit 12-digit MAC address (using lower case hex digits). For example, 000e08abcdef.
- $SN expands to the unit serial number. For example, 88012BA01234.

Other values can be macro expanded in this way, including all the general purpose parameters, GPP_A through GPP_P. An example of this process can be seen in TFTP Resync, on page 1. Macro expansion is not limited to the URL file name, but can also be applied to any portion of the profile rule parameter. These parameters are referenced as $A through $P. For a complete list of variables that are available for macro expansion, see Macro Expansion Variables.

In this exercise, a profile specific to a phone is provisioned on a TFTP server.

Provision a Specific IP Phone Profile on a TFTP Server

Procedure

Step 1 Obtain the MAC address of the phone from its product label. (The MAC address is the number, using numbers and lower–case hex digits, such as 000e08aabbcc.

Step 2 Copy the `basic.txt` configuration file (described in TFTP Resync, on page 1) to a new file named `ataxxxx.cfg` (replacing `xxxx` with the `macaddress` with the MAC address of the phone).

Step 3 Move the new file in the virtual root directory of the TFTP server.

Step 4 Access the phone administration web page. See Access the Phone Web Interface.

Step 5 Select **Voice** > **Provisioning**.

Step 6 Enter `tftp://192.168.1.200/ata$MA.cfg` in the **Profile Rule** field.

```
<Profile_Rule>
  tftp://192.168.1.200/ata$MA.cfg
</Profile_Rule>
```

Step 7 Click **Submit All Changes**. This causes an immediate reboot and resync.

When the next resync occurs, the phone retrieves the new file by expanding the $MA macro expression into its MAC address.

HTTP GET Resync

HTTP provides a more reliable resync mechanism than TFTP because HTTP establishes a TCP connection and TFTP uses the less reliable UDP. In addition, HTTP servers offer improved filtering and logging features compared to TFTP servers.

On the client side, the phone does not require any special configuration setting on the server to be able to resync by using HTTP. The Profile_Rule parameter syntax for using HTTP with the GET method is similar to the syntax that is used for TFTP. If a standard web browser can retrieve a profile from your HTTP server, the phone should be able to do so as well.

Resync with HTTP GET

Procedure

Step 1 Install an HTTP server on the local PC or other accessible host.

The open source Apache server can be downloaded from the internet.

Step 2 Copy the `basic.txt` configuration profile (described in TFTP Resync, on page 1) onto the virtual root directory of the installed server.

Step 3 To verify proper server installation and file access to `basic.txt`, access the profile with a web browser.

Step 4 Modify the Profile_Rule of the test phone to point to the HTTP server in place of the TFTP server, so as to download its profile periodically.

For example, assuming the HTTP server is at 192.168.1.300, enter the following value:

```
<Profile_Rule>
http://192.168.1.200/basic.txt
</Profile_Rule>
```

Step 5 Click **Submit All Changes**. This causes an immediate reboot and resync.

Step 6 Observe the syslog messages that the phone sends. The periodic resyncs should now be obtaining the profile from the HTTP server.

Step 7 In the HTTP server logs, observe how information that identifies the test phone appears in the log of user agents.

This information should include the manufacturer, product name, current firmware version, and serial number.

Provisioning Through Cisco XML

For each of the phones, designated as xxxx here, you can provision through Cisco XML functions.

You can send an XML object to the phone by a SIP Notify packet or an HTTP Post to the CGI interface of the phone: `http://IPAddressPhone/CGI/Execute`.

The CP-xxxx-3PCC extends the Cisco XML feature to support provisioning via an XML object:

```
<CP-xxxx-3PCCExecute>
        <ExecuteItem URL=Resync:[profile-rule]/>
</CP-xxxx-3PCCExecute>
```

After the phone receives the XML object, it downloads the provisioning file from [profile-rule]. This rule uses macros to simplify the development of the XML services application.

URL Resolution with Macro Expansion

Subdirectories with multiple profiles on the server provide a convenient method for managing a large number of deployed devices. The profile URL can contain:

- A provisioning server name or an explicit IP address. If the profile identifies the provisioning server by name, the phone performs a DNS lookup to resolve the name.
- A nonstandard server port that is specified in the URL by using the standard syntax `:port` following the server name.
- The subdirectory of the server virtual root directory where the profile is stored, specified by using standard URL notation and managed by macro expansion.

For example, the following Profile_Rule requests the profile file ($PN.cfg), in the server subdirectory `/cisco/config`, from the TFTP server that is running on host prov.telco.com listening for a connection on port 6900:

```
<Profile_Rule>
tftp://prov.telco.com:6900/cisco/config/$PN.cfg
</Profile_Rule>
```

A profile for each phone can be identified in a general purpose parameter, with its value referred within a common profile rule by using macro expansion.

For example, assume GPP_B is defined as `Dj6Lmp23Q`.

The Profile_Rule has the value:

```
tftp://prov.telco.com/cisco/$B/$MA.cfg
```

When the device resyncs and the macros are expanded, the phone with a MAC address of 000e08012345 requests the profile with the name that contains the device MAC address at the following URL:

```
tftp://prov.telco.com/cisco/Dj6Lmp23Q/000e08012345.cfg
```

Secure HTTPS Resync

These mechanisms are available on the phone for resyncing by using a secure communication process:

- Basic HTTPS Resync
- HTTPS with Client Certificate Authentication
- HTTPS Client Filtering and Dynamic Content

Basic HTTPS Resync

HTTPS adds SSL to HTTP for remote provisioning so that the:

- The phone can authenticate the provisioning server.
- Provisioning server can authenticate the phone.
- Confidentiality of information exchanged between the phone and the provisioning server is ensured.

SSL generates and exchanges secret (symmetric) keys for each connection between the phone and the server, using public/private key pairs that are pre-installed in the phone and the provisioning server.

On the client side, the phone does not require any special configuration setting on the server to be able to resync using HTTPS. The Profile_Rule parameter syntax for using HTTPS with the GET method is similar to the syntax that is used for HTTP or TFTP. If a standard web browser can retrieve a profile from a your HTTPS server, the phone should be able to do so as well.

In addition to installing a HTTPS server, a SSL server certificate that Cisco signs must be installed on the provisioning server. The devices cannot resync to a server that is using HTTPS unless the server supplies a Cisco-signed server certificate. Instructions for creating signed SSL Certificates for Voice products can be found at https://supportforums.cisco.com/docs/DOC-9852.

Related Topics

Secure HTTPS Resync, on page 7

Authenticate with Basic HTTPS Resync

Procedure

Step 1 Install an HTTPS server on a host whose IP address is known to the network DNS server through normal hostname translation.

The open source Apache server can be configured to operate as an HTTPS server when installed with the open source mod_ssl package.

Step 2 Generate a server Certificate Signing Request for the server. For this step, you might need to install the open source OpenSSL package or equivalent software. If using OpenSSL, the command to generate the basic CSR file is as follows:

```
openssl req -new -out provserver.csr
```

This command generates a public/private key pair, which is saved in the `privkey.pem` file.

Step 3 Submit the CSR file (provserver.csr) to Cisco for signing.

A signed server certificate is returned (provserver.cert) along with a Sipura CA Client Root Certificate, spacroot.cert.

See https://supportforums.cisco.com/docs/DOC-9852 for more information

Step 4 Store the signed server certificate, the private key pair file, and the client root certificate in the appropriate locations on the server.

In the case of an Apache installation on Linux, these locations are typically as follows:

```
# Server Certificate:
SSLCertificateFile /etc/httpd/conf/provserver.cert
# Server Private Key:
SSLCertificateKeyFile /etc/httpd/conf/pivkey.pem
# Certificate Authority:
SSLCACertificateFile /etc/httpd/conf/spacroot.cert
```

Step 5 Restart the server.

Step 6 Copy the `basic.txt` configuration file (described in TFTP Resync, on page 1) onto the virtual root directory of the HTTPS server.

Step 7 Verify proper server operation by downloading `basic.txt` from the HTTPS server by using a standard browser from the local PC.

Step 8 Inspect the server certificate that the server supplies.

The browser probably does not recognize the certificate as valid unless the browser has been pre-configured to accept Cisco as a root CA. However, the phones expect the certificate to be signed this way.

Modify the Profile_Rule of the test device to contain a reference to the HTTPS server, for example:

```
<Profile_Rule>
https://my.server.com/basic.txt
</Profile_Rule>
```

This example assumes the name of the HTTPS server is **my.server.com**.

Step 9 Click **Submit All Changes**.

Step 10 Observe the syslog trace that the phone sends.

The syslog message should indicate that the resync obtained the profile from the HTTPS server.

Step 11 (Optional) Use an Ethernet protocol analyzer on the phone subnet to verify that the packets are encrypted.

In this exercise, client certificate verification was not enabled. The connection between the phone and server is encrypted. However, the transfer is not secure because any client can connect to the server and request the file, given knowledge of the file name and directory location. For secure resync, the server must also authenticate the client, as demonstrated in the exercise described in HTTPS with Client Certificate Authentication, on page 9.

HTTPS with Client Certificate Authentication

In the factory default configuration, the server does not request an SSL client certificate from a client. Transfer of the profile is not secure because any client can connect to the server and request the profile. You can edit the configuration to enable client authentication; the server requires a client certificate to authenticate the phone before it accepts a connection request.

Because of this requirement, the resync operation cannot be independently tested by using a browser that lacks the proper credentials. The SSL key exchange within the HTTPS connection between the test phone and the server can be observed with the ssldump utility. The utility trace shows the interaction between client and server.

Related Topics

Secure HTTPS Resync, on page 7

Authenticate HTTPS with Client Certificate

Procedure

Step 1 Enable client certificate authentication on the HTTPS server.

Step 2 In Apache (v.2), set the following in the server configuration file:

```
SSLVerifyClient  require
```

Also, ensure that the spacroot.cert has been stored as shown in the Basic HTTPS Resync, on page 7 exercise.

Step 3 Restart the HTTPS server and observe the syslog trace from the phone.

Each resync to the server now performs symmetric authentication, so that both the server certificate and the client certificate are verified before the profile is transferred.

Step 4 Use ssldump to capture a resync connection between the phone and the HTTPS server.

If client certificate verification is properly enabled on the server, the ssldump trace shows the symmetric exchange of certificates (first server-to-client, then client-to-server) before the encrypted packets that contain the profile.

With client authentication enabled, only a phone with a MAC address that matches a valid client certificate can request the profile from the provisioning server. The server rejects a request from an ordinary browser or other unauthorized device.

Configure a HTTPS Server for Client Filtering and Dynamic Content

If the HTTPS server is configured to require a client certificate, the information in the certificate identifies the resyncing phone and supplies it with the correct configuration information.

The HTTPS server makes the certificate information available to CGI scripts (or compiled CGI programs) that are invoked as part of the resync request. For the purpose of illustration, this exercise uses the open source Perl scripting language, and assumes that Apache (v.2) is used as the HTTPS server.

Procedure

Step 1 Install Perl on the host that is running the HTTPS server.

Step 2 Generate the following Perl reflector script:

```
#!/usr/bin/perl -wT
use strict;
print "Content-Type: text/plain\n\n";
print "<flat-profile><GPP_D>";

print "OU=$ENV{'SSL_CLIENT_I_DN_OU'},\n";
print "L=$ENV{'SSL_CLIENT_I_DN_L'},\n";
print "S=$ENV{'SSL_CLIENT_I_DN_S'}\n";
print "</GPP_D></flat-profile>";
```

Step 3 Save this file with the file name `reflect.pl`, with executable permission (chmod 755 on Linux), in the CGI scripts directory of the HTTPS server.

Step 4 Verify accessibility of CGI scripts on the server (that is, `/cgi-bin/...`).

Step 5 Modify the Profile_Rule on the test device to resync to the reflector script, as in the following example:

```
https://prov.server.com/cgi-bin/reflect.pl?
```

Step 6 Click **Submit All Changes**.

Step 7 Observe the syslog trace to ensure a successful resync.

Step 8 Access the phone administration web page. See Access the Phone Web Interface.

Step 9 Select **Voice** > **Provisioning**.

Step 10 Verify that the GPP_D parameter contains the information that the script captured.

This information contains the product name, MAC address, and serial number if the test device carries a unique certificate from the manufacturer. The information contains generic strings if the unit was manufactured before firmware release 2.0.

A similar script can determine information about the resyncing device and then provide the device with appropriate configuration parameter values.

HTTPS Certificates

The phone provides a reliable and secure provisioning strategy that is based on HTTPS requests from the device to the provisioning server. Both a server certificate and a client certificate are used to authenticate the phone to the server and the server to the phone.

In addition to Cisco issued certifications, the phone also accepts server certificates from a set of commonly used SSL certificate providers.

To use HTTPS with the phone, you must generate a Certificate Signing Request (CSR) and submit it to Cisco. The phone generates a certificate for installation on the provisioning server. The phone accepts the certificate when it seeks to establish an HTTPS connection with the provisioning server.

HTTPS Methodology

HTTPS encrypts the communication between a client and a server, thus protecting the message contents from other network devices. The encryption method for the body of the communication between a client and a server is based on symmetric key cryptography. With symmetric key cryptography, a client and a server share a single secret key over a secure channel that is protected by Public/Private key encryption.

Messages encrypted by the secret key can only be decrypted by using the same key. HTTPS supports a wide range of symmetric encryption algorithms. The phone implements up to 256-bit symmetric encryption, using the American Encryption Standard (AES), in addition to 128-bit RC4.

HTTPS also provides for the authentication of a server and a client engaged in a secure transaction. This feature ensures that a provisioning server and an individual client cannot be spoofed by other devices on the network. This capability is essential in the context of remote endpoint provisioning.

Server and client authentication is performed by using public/private key encryption with a certificate that contains the public key. Text that is encrypted with a public key can be decrypted only by its corresponding private key (and vice versa). The phone supports the Rivest-Shamir-Adleman (RSA) algorithm for public/private key cryptography.

SSL Server Certificate

Each secure provisioning server is issued a secure sockets layer (SSL) server certificate that Cisco signs directly. The firmware that runs on the phone recognizes only a Cisco certificate as valid. When a client connects to a server by using HTTPS, it rejects any server certificate that is not signed by Cisco.

This mechanism protects the service provider from unauthorized access to the phone, or any attempt to spoof the provisioning server. Without such protection, an attacker might be able to reprovision the phone, to gain configuration information, or to use a different VoIP service. Without the private key that corresponds to a valid server certificate, the attacker is unable to establish communication with a phone.

Obtain a Server Certificate

Procedure

Step 1 Contact a Cisco support person who will work with you on the certificate process. If you are not working with a specific support person, email your request to ciscosb-certadmin@cisco.com.

Step 2 Generate a private key that will be used in a CSR (Certificate Signing Request). This key is private and you do not need to provide this key to Cisco support. Use open source "openssl" to generate the key. For example:

```
openssl genrsa -out <file.key> 1024
```

Step 3 Generate a CSR that contains fields that identify your organization and location. For example:

```
openssl req -new -key <file.key> -out <file.csr>
```

You must have the following information:

- Subject field—Enter the Common Name (CN) that must be an FQDN (Fully Qualified Domain Name) syntax. During SSL authentication handshake, the phone verifies that the certificate it receives is from the machine that presented it.
- Server hostname—For example, provserv.domain.com.
- Email address—Enter an email address so that customer support can contact you if needed. This email address is visible in the CSR.

Step 4 Email the CSR (in zip file format) to the Cisco support person or to ciscosb-certadmin@cisco.com. The certificate is signed by Cisco. Cisco sends the certificate to you to install on your system.

Client Certificate

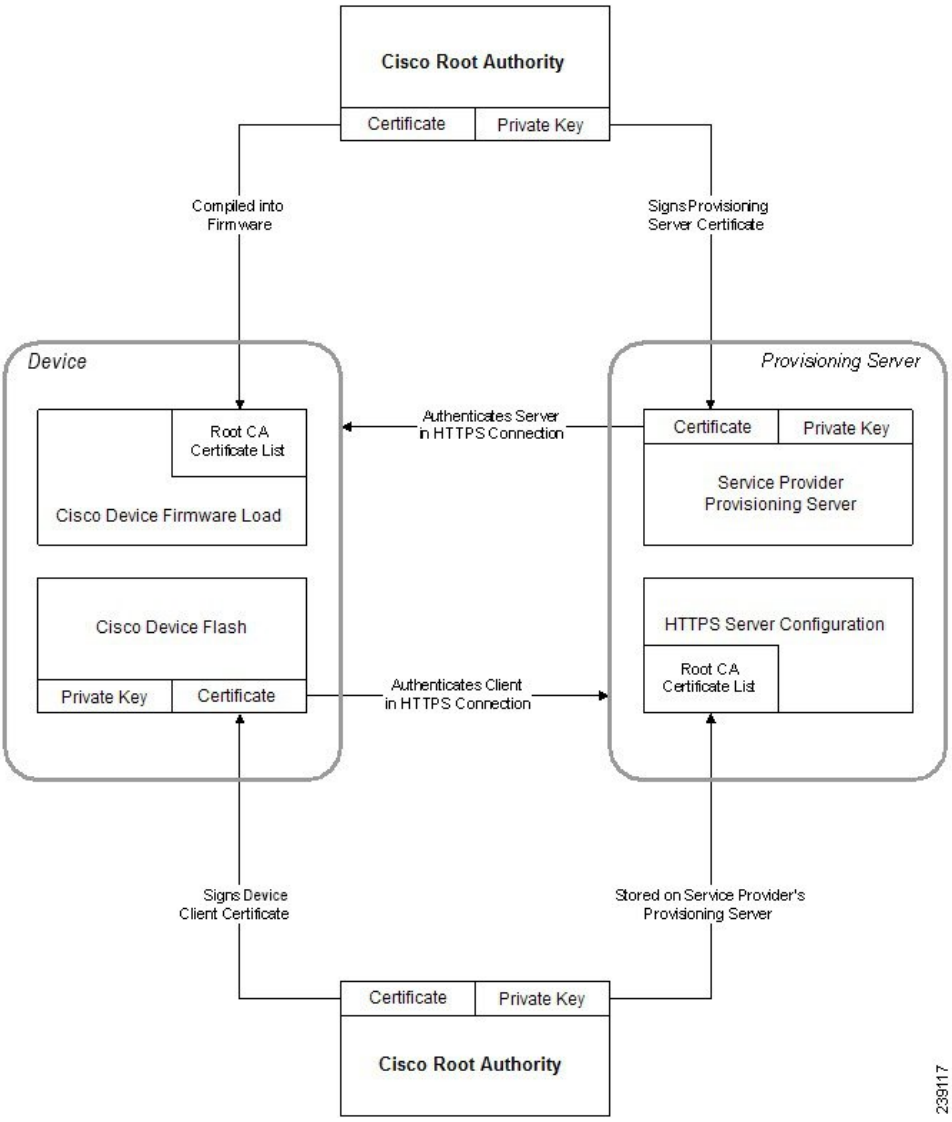
In addition to a direct attack on a phone, an attacker might attempt to contact a provisioning server through a standard web browser or another HTTPS client to obtain the configuration profile from the provisioning server. To prevent this kind of attack, each phone also carries a unique client certificate, signed by Cisco, that includes identifying information about each individual endpoint. A certificate authority root certificate that is capable of authenticating the device client certificate is given to each service provider. This authentication path allows the provisioning server to reject unauthorized requests for configuration profiles.

Certificate Structure

The combination of a server certificate and a client certificate ensures secure communication between a remote phone and its provisioning server. The figure below illustrates the relationship and placement of certificates, public/private key pairs, and signing root authorities, among the Cisco client, the provisioning server, and the certification authority.

The upper half of the diagram shows the Provisioning Server Root Authority that is used to sign the individual provisioning server certificate. The corresponding root certificate is compiled into the firmware, which allows the phone to authenticate authorized provisioning servers.

Figure 1: Certificate Authority Flow



Configure a Custom Certificate Authority

Digital certificates can be used to authenticate network devices and users on the network. They can be used to negotiate IPSec sessions between network nodes.

A third party uses a Certificate Authority certificate to validate and authenticate two or more nodes that are attempting to communicate. Each node has a public and private key. The public key encrypts data. The private key decrypts data. Because the nodes have obtained their certificates from the same source, they are assured of their respective identities.

The device can use digital certificates provided by a third-party Certificate Authority (CA) to authenticate IPSec connections.

The phones support a set of preloaded Root Certificate Authority embedded in the firmware:

- Cisco Small Business CA Certificate

- CyberTrust CA Certificate
- Verisign CA certificate
- Sipura Root CA Certificate
- Linksys Root CA Certificate

Before you begin

Access the phone administration web page. See Access the Phone Web Interface.

Procedure

Step 1 Select **Info** > **Status**.

Step 2 Scroll to **Custom CA Status** and see the following fields:

- Custom CA Provisioning Status—Indicates the provisioning status.
  - Last provisioning succeeded on mm/dd/yyyy HH:MM:SS; or
  - Last provisioning failed on mm/dd/yyyy HH:MM:SS
- Custom CA Info—Displays information about the custom CA.
  - Installed—Displays the "CN Value," where "CN Value" is the value of the CN parameter for the Subject field in the first certificate.
  - Not Installed—Displays if no custom CA certificate is installed.

Profile Management

This section demonstrates the formation of configuration profiles in preparation for downloading. To explain the functionality, TFTP from a local PC is used as the resync method, although HTTP or HTTPS can be used as well.

Compress an Open Profile with Gzip

A configuration profile in XML format can become quite large if the profile specifies all parameters individually. To reduce the load on the provisioning server, the phone supports compression of the XML file, by using the deflate compression format that the gzip utility (RFC 1951) supports.

Note Compression must precede encryption for the phone to recognize a compressed and encrypted XML profile.

For integration into customized back-end provisioning server solutions, the open source zlib compression library can be used in place of the standalone gzip utility to perform the profile compression. However, the phone expects the file to contain a valid gzip header.

Procedure

Step 1 Install gzip on the local PC.

Step 2 Compress the `basic.txt` configuration profile (described in TFTP Resync, on page 1) by invoking gzip from the command line:

```
gzip basic.txt
```

This generates the deflated file `basic.txt.gz`.

Step 3 Save the `basic.txt.gz` file in the TFTP server virtual root directory.

Step 4 Modify the Profile_Rule on the test device to resync to the deflated file in place of the original XML file, as shown in the following example:

```
tftp://192.168.1.200/basic.txt.gz
```

Step 5 Click **Submit All Changes.**

Step 6 Observe the syslog trace from the phone.

Upon resync, the phone downloads the new file and uses it to update its parameters.

Related Topics

Open Profile Compression

Encrypt a Profile with OpenSSL

A compressed or uncompressed profile can be encrypted (however, a file must be compressed before it is encrypted). Encryption is useful when the confidentiality of the profile information is of particular concern, such as when TFTP or HTTP is used for communication between the phone and the provisioning server.

The phone supports symmetric key encryption by using the 256-bit AES algorithm. This encryption can be performed by using the open source OpenSSL package.

Procedure

Step 1 Install OpenSSL on a local PC. This might require that the OpenSSL application be recompiled to enable AES.

Step 2 Using the `basic.txt` configuration file (described in TFTP Resync, on page 1), generate an encrypted file with the following command:

```
>openssl enc -aes-256-cbc -k MyOwnSecret -in basic.txt -out basic.cfg
```

The compressed `basic.txt.gz` file that was created in Compress an Open Profile with Gzip, on page 14 also can be used, because the XML profile can be both compressed and encrypted.

Step 3 Store the encrypted `basic.cfg` file in the TFTP server virtual root directory.

Step 4 Modify the Profile_Rule on the test device to resync to the encrypted file in place of the original XML file. The encryption key is made known to the phone with the following URL option:

```
[--key MyOwnSecret ] tftp://192.168.1.200/basic.cfg
```

Step 5 Click **Submit All Changes**.

Step 6 Observe the syslog trace from the phone.

Upon resync, the phone downloads the new file and uses it to update its parameters.

Related Topics

AES-256-CBC Encryption

Create Partitioned Profiles

A phone downloads multiple separate profiles during each resync. This practice allows management of different kinds of profile information on separate servers and maintenance of common configuration parameter values that are separate from account specific values.

Procedure

Step 1 Create a new XML profile, `basic2.txt`, that specifies a value for a parameter that makes it distinct from the earlier exercises. For instance, to the `basic.txt` profile, add the following:

```
<GPP_B>ABCD</GPP_B>
```

Step 2 Store the `basic2.txt` profile in the virtual root directory of the TFTP server.

Step 3 Leave the first profile rule from the earlier exercises in the folder, but configure the second profile rule (Profile_Rule_B) to point to the new file:

```
<Profile_Rule_B>tftp://192.168.1.200/basic2.txt
</Profile_Rule_B>
```

Step 4 Click **Submit All Changes**.

The phone now resyncs to both the first and second profiles, in that order, whenever a resync operation is due.

Step 5 Observe the syslog trace to confirm the expected behavior.

Manage Provisioning with Parameter Name Aliases

When generating an XML profile for the ATA, it might be convenient to assign names to certain configuration parameters that are different from the canonical names recognized by the ATA. For example, a customer account database might generate XML element tags for a customer telephone number and SIP registration password with names, such as SIP-number and SIP-password. These names can be mapped to the canonical names (User_ID_1_ and Password_1_) before being applied to Line1.

In many instances, the back-end provisioning solution used by the service provider can perform this mapping. However, the ATA itself can remap the parameter names internally. To do this, an alias map is defined and stored in one of the general purpose provisioning parameters. Then, the profile rule which invokes the resync is directed to remap the non-canonical XML elements as specified by the alias map.

Procedure

Step 1 Generate a profile named customer.XML containing the proprietary customeraccount XML form indicated in the following example:

```
<customer-account>
<SIP-number> 17775551234</SIP-number>
<SIP-password> 512835907884</SIP-password>
</customer-account>
```

Step 2 Store the profile in the TFTP server virtual root directory.

Step 3 Open the web interface on the device to **Voice** > **Provisioning**, and edit GPP_A to contain the alias map. Do not enter new lines through the web interface, instead simply enter each alias consecutively:

```
/customer-account/SIP-number = /flat-profile/User_ID_1_ ;
/customer-account/SIP-password = /flat-profile/Password_1_ ;
```

Step 4 Edit the Profile_Rule to point to the new XML profile, and specify the alias map as a URL option, as follows:

```
[--alias a ] tftp://192.168.1.200/customer.xml
```

Step 5 Click **Submit All Changes**.

When the ATA resyncs, it receives the XML profile, remaps the elements, as indicated by the alias map, and populates the User_ID_1_ and Password_1_ parameters.

Step 6 View the Line 1 tab to verify the new configuration.

Note The ATA supports alias remapping of a limited number of parameters. It is not meant to rename all parameters in its configuration.