Protector and TLS Enabled Postfix for Explicit MTA

Protector and TLS Enabled Postfix for Explicit MTA | TRITON AP-DATA | v8.3.x | 15-Dec-2016

The protector component in Forcepoint™ TRITON® AP-DATA v8.3.x includes a Postfix release compiled with Transport Layer Security (TLS) support. Configuring Postfix allows you to enable TLS support. Although TLS is not officially supported, Postfix is available to allow for individual customer configurations.

This document provides a simple example TLS configuration for use as a test case and as a reference for future deployments. The sample configuration is stored on the protector in /etc/postfix/main.cf. Note that every time you modify this file, you must reload Postfix using one of the following operations:

- **postfix** reload
- **postfix**-reconf
- Clicking **Deploy** in Data Security manager

In the protector, Postfix serves as a store-and-forward proxy. This means that it functions as both a server (getting messages from the previous hop) and a client (delivering the non-blocked messages to the next hop).

Because previous and next hops may have different TLS requirements, you configure TLS settings for server and client modes differently.

Client Side

In this example, assume at least some *next* hops require TLS:

1. On the protector, open /etc/postfix/tls_policy in a text editor. If this file does not exist on your machine, create one.
2. Add this line to the file:

   next.hop.domain encrypt
3. Compile the file using the following command:

   ```
   postmap hash:/etc/postfix/tls_policy
   ```
4. Open /etc/postfix/main.cf in a text editor.
5. Add these lines to the file:

   relayhost = next.hop.domain ## maintained by management !!!

```
## certs files
smtp_tls_cert_file=/etc/pki/tls/certs/
mydomain.com.cert
smtp_tls_key_file=/etc/pki/tls/private/
mydomain.com.key
smtp_tls_CAfile=/etc/pki/tls/cert.pem
## policy map
smtp_tls_policy_maps = hash:/etc/postfix/tls_policy
```

6. Run the following command:

   postfix reload

Notes:

- The policy map is required for better granularity. Specifically, Postfix also traverses messages internally (for example, to the pamad plugin) and these transfers should be plain.
- Certificates are very customer-dependent. For this example, 3 PEM files were created:
  - mydomain.com.cert (client certificate, not necessarily required by servers)
  - mydomain.com.key (client's private key, sometimes included in client certificate)
  - default cert.pem (Certificate Authority (CA) certificate).

  For this sample we've executed the following command:

  ```
  genkey -days 265 $(hostname)
  ```

  The **genkey** utility is a part of crypto utilities package, which is not installed by default and should be installed manually or using "yum." To install the package with "yum", type the following command:

  ```
  yum install crypto-utils
  ```

  If packages installation isn't an option, you can create certificates using the **openssl** command:

  ```
  openssl req -new -nodes -keyout myhost.com.key -out
  req.pem
  ```

  In this case, sign req.pem with a Certificate Authority and get in return the myhost.com.cert and cert.pem.

Server Side

Now assume at least some *previous* hops require TLS:

1. Open /etc/postfix/main.cf in a text editor.
2. Add these lines to the file:

```
smtpd_tls_cert_file= /opt/websense/PolicyEngine/
allcerts.cer
smtpd_tls_key_file= $smtpd_tls_cert_file
smtpd_tls_security_level = may
```

3. Run the following command:

   ```
   postfix reload
   ```

Notes:

- In this sample we use protector's certificates. Some clients (previous hops) may require this certificate to be trusted by a known CA.
- Optionally, private key is included in the certificate file.
- Security level - in this sample, TLS is enabled but not mandatory. This can be changed.

For further details, see: http://www.postfix.org/TLS_README.html and http://www.postfix.org/postconf.5.html.