X.509 CERTIFICATE BASED AUTHENTICATION FOR NETCONF AND RESTCONF

tail-f a Cisco company

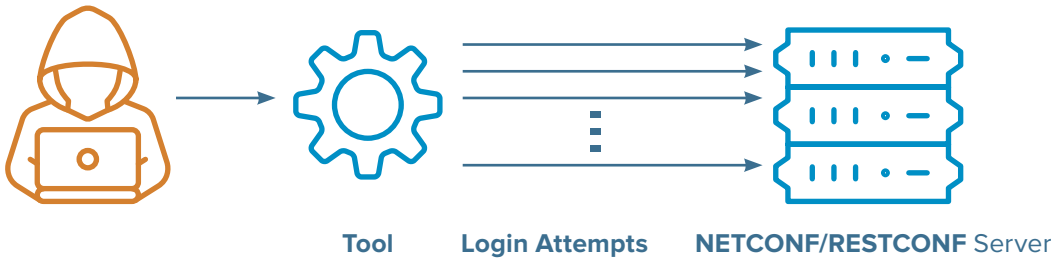
Table of Contents

1. Overview

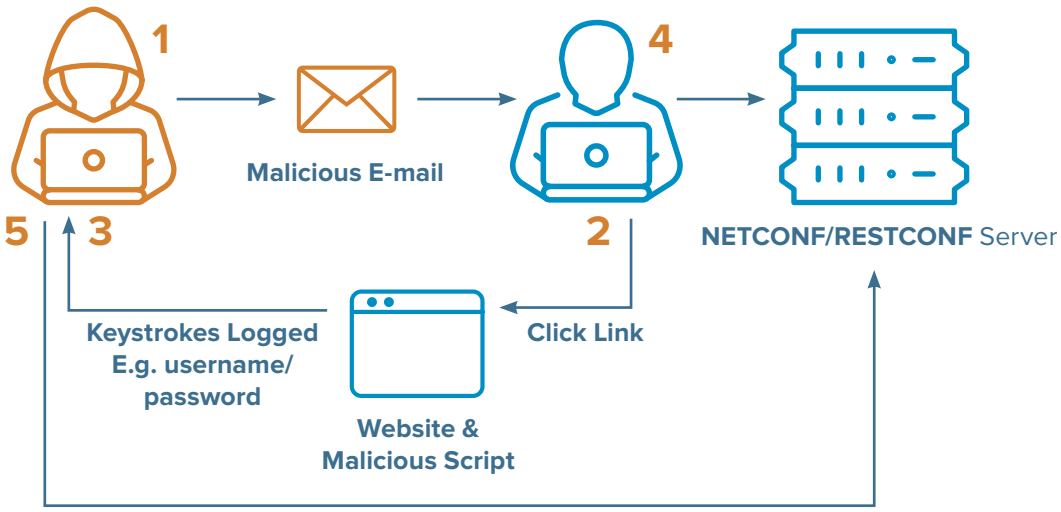
In the modern world of communication where the Internet plays an obviously vital role, servers need to verify the client's identity to prevent malicious clients from gaining access to read or change the server content. Clients need to be able to trust that the servers in order to prevent malicious servers from providing false content and gaining access to content written by the client to the server.

A common but very basic way for servers to authenticate client users is for the client to identify itself by providing a username and password to the server. Another common way for servers to authenticate clients is to use public-key authentication. After authenticating who the client is, the server uses the client's identity to determine the client's level of access to the server, i.e. service authorization.

If the server requires the client to use very strong passwords, the server is at least safe from basic brute force attacks. The same goes for public-key authentication with strong encryption.



To prevent attackers from using sniffing tools to intercept unencrypted usernames and passwords communicated over the network, a pre-shared key between the client and server can be used to encrypt the username and password, but it is more common is to use a non-authenticated key-agreement protocol, such as Diffie-Hellman, where the client and server can influence and agree on a key used to encrypt the username and password or the public key that the client shares with the server. However, this does not protect the server from malicious clients that have in some other way obtained the secret username and password or the public key.



A common way for clients to authenticate the servers is to store the public key used by the server together with the server hostname or IP address in a known hosts file. Then, at least after the first time the client connects to the server, the server can be authenticated by the client to prevent, for example, man-in-the-middle attacks where the attacker secretly relays and possibly alters the communications between two parties who believe that they are directly communicating with each other. However, this does not protect the client from malicious servers the first time the client connects to the server or if the malicious server has somehow obtained the intended servers public key.

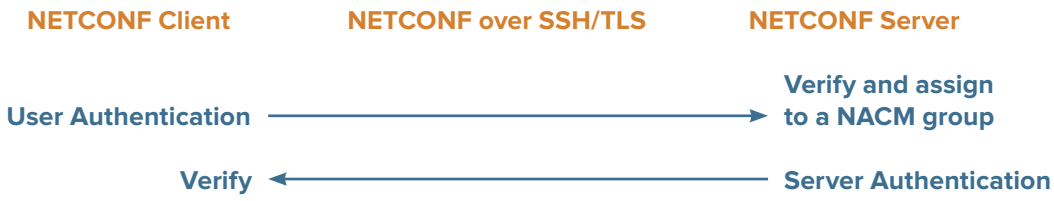
User **Man in the Middle** **NETCONF/RESTCONF** Server

In a distributed system, there are different components that need to communicate with each other. In a cloud-native world, those components will likely be containers exchanging messages with each other or with other internal or external components. In this application note, we explore how the NETCONF [RFC6241] and RESTCONF [RFC 8040] protocols over TLS [RFC 7589] and SSH [RFC 6242] can be secured using X.509 and the Internet Public Key Infrastructure (PKIX) [RFC 5280] so that only legitimate users will be allowed to access the intended services. We demonstrate how NETCONF and RESTCONF clients and servers can be set up to identify themselves to each other and set up secure connections between them so that malicious components can't get involved in these communications.

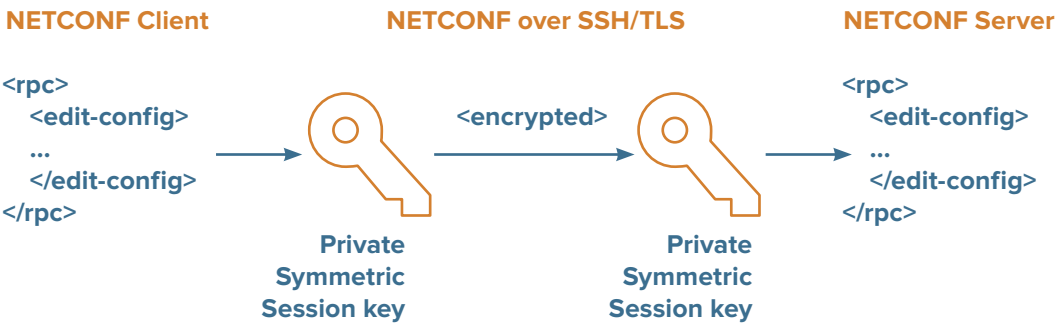
2. Secure Connections

There are two parts to setting up a secure connection between a NETCONF or RESTCONF client and server:

- First, the NETCONF or RESTCONF client and server authenticate each other to ensure they are not accessing malicious servers or providing access to malicious clients.
  - The server wants to authenticate the client's identity, match it to a Network Configuration Access Control Model (NACM) [RFC 8341] user group, and authorize the client's NETCONF or RESTCONF operations against access control rules.
  - The client wants to verify that the server's identity is the intended one.



  - The second part is the encryption. When the NETCONF or RESTCONF client and server are communicating, you don't want any third parties to be able to intercept (or worse, interfere with) that communication channel.



3. Internet X.509 Public Key Infrastructure Certificate (PKIX) and Certificate Revocation List (CRL) Profile

The ITU Telecommunication Standardization Sector (ITU-T) recommendation X.509 is a portion of the X.500 series of standards describing directory services. An “X.509 certificate” is usually short for IETF's PKIX and CRL profile of the X.509 certificate standard.

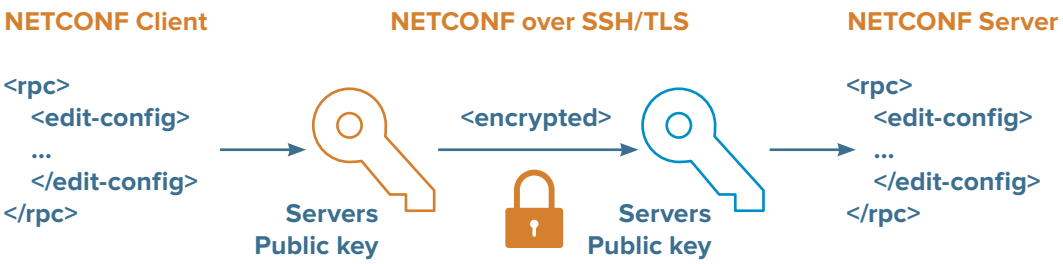
Public and Private Keys

The ‘P’ and ‘K’ in PKIX are referring to a cryptographic system that uses pairs of keys:

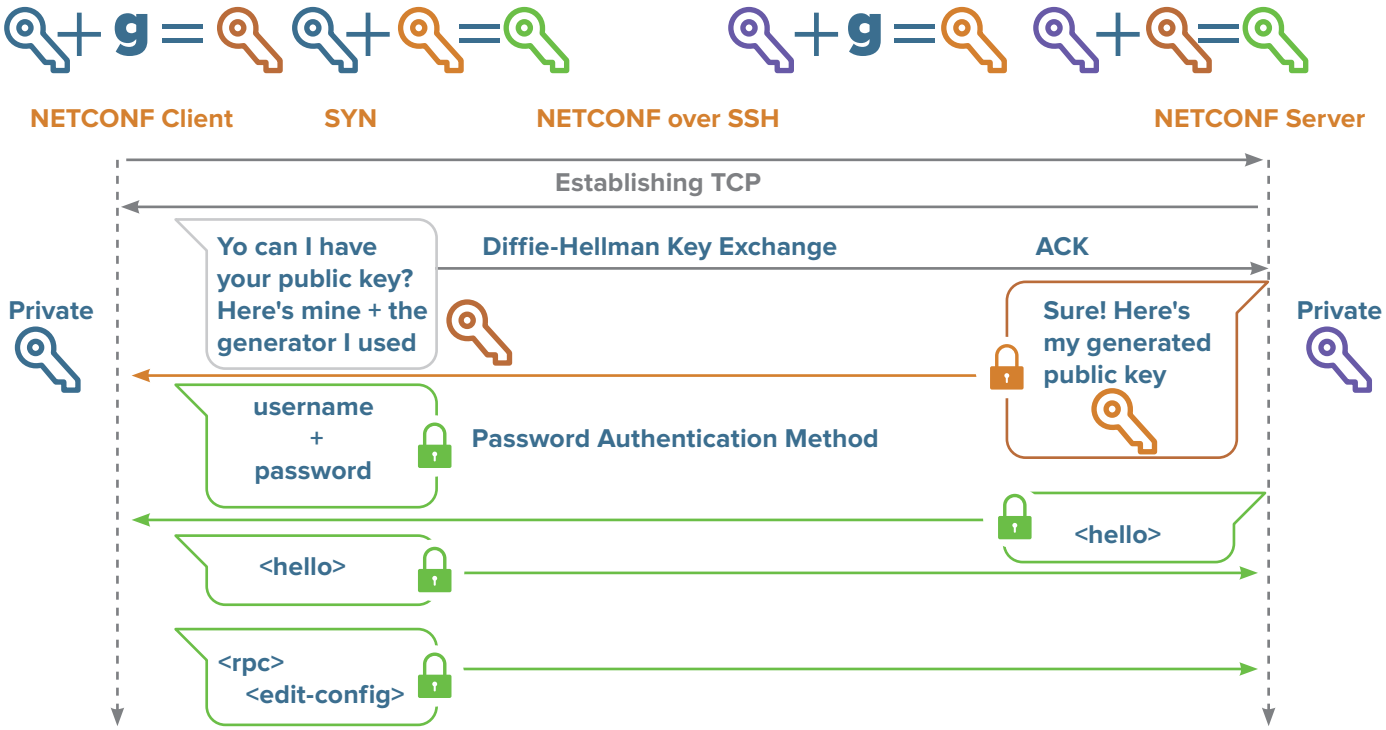
- Public keys, which may be spread to many.
- Private keys, which are known only to the owner.

A public key, such as the one from the X.509 certificate, can be used to encrypt a message or digital signature that can only be decrypted by the holder of the corresponding private key.

Each key pair is unique and the two keys work together. If you have the private key, you can prove you have it without showing what it is. It's like proving you know a password without having to show someone the password.



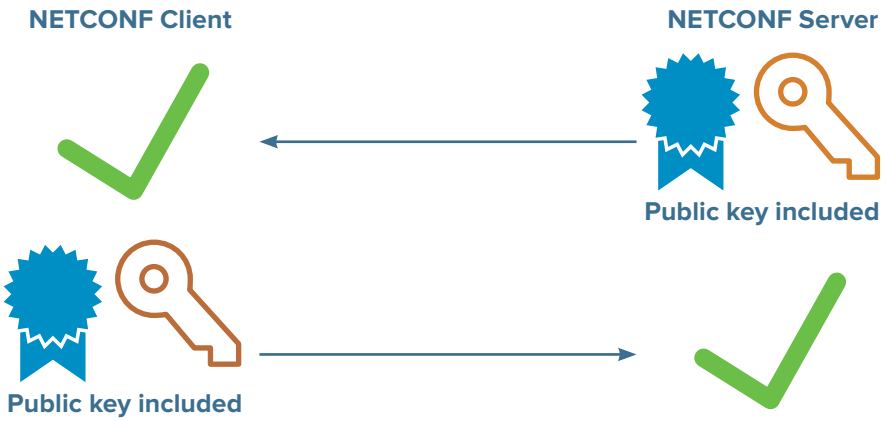
A NETCONF or RESTCONF client and server use their private key and their public key from the X.509 certificate in combination for both mutual authentication using a digital signature and for the encryption part of setting up the X.509 certificate-based secure connection between them.



Public Key Infrastructure

The ´I´ in PK**I**X stands for **i**nfrastructure and binds public keys with an identity. NETCONF and RESTCONF clients and servers using a public key require confidence that the associated private key is owned by the correct remote client or server. The binding is asserted by a trusted Certificate Authority (CA) digitally signing each certificate. The mechanism for distribution and binding of public keys is known as a Public Key Infrastructure (PKI), and the most widely used one is X.509 certificates.

The certificate is itself signed by a Certificate Authority (CA) using the CA's private key. This verifies the authenticity of the certificate.



X.509v3

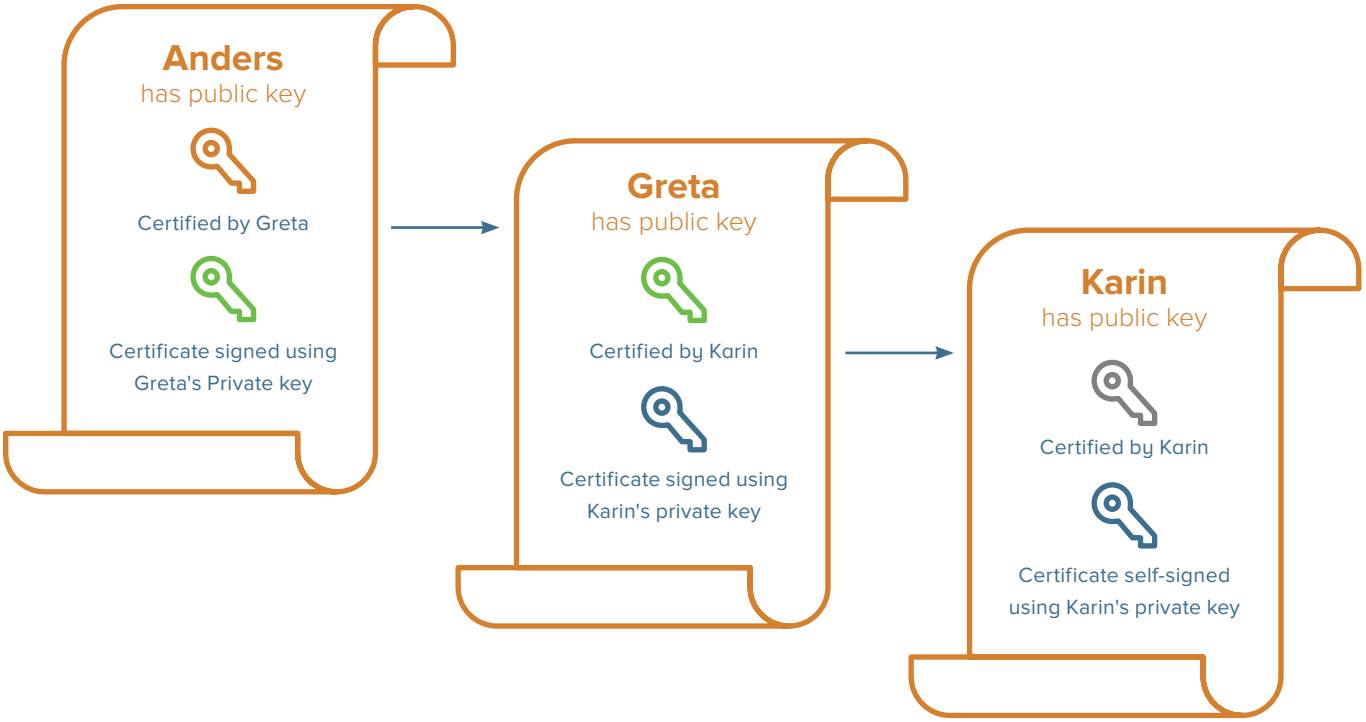
The ‘X’ in PKI**X** stands for **X**.509v3 and defines the format of the certificate used to distribute the public key. An X.509 certificate has a CA signature and limited valid lifetime that can be used independently for mutual client-server authentication. X.509 certificates can, therefore, be distributed via untrusted communications and server systems and can be cached in unsecured storage in the NETCONF or RESTCONF client and server’s file system or ConfD’s CDB.

Certificate Authority

A Certificate Authority (CA) is a trusted entity that signs a certificate that you use with your NETCONF or RESTCONF client or server. This signing verifies that the identity and information contained in the certificate is correct. You should therefore only trust a certificate that has been signed by a CA you trust.

The CA that signs a certificate does not have to be a root CA that issues what is a called a self-signed certificate (an X.509 certificate the root CA signed for itself). It is ok if the identity represented by the certificate is the same as the identity whose private key is used to sign the certificate. If you can trust that identity, you can trust the certificate.

To tell the CA about the X.509 certificate you want to be generated, you need a Certificate Signing Request (CSR) that you can generate using tools like openssl or certtool and send to the CA to request an X.509 certificate.



Certificate Revocation List

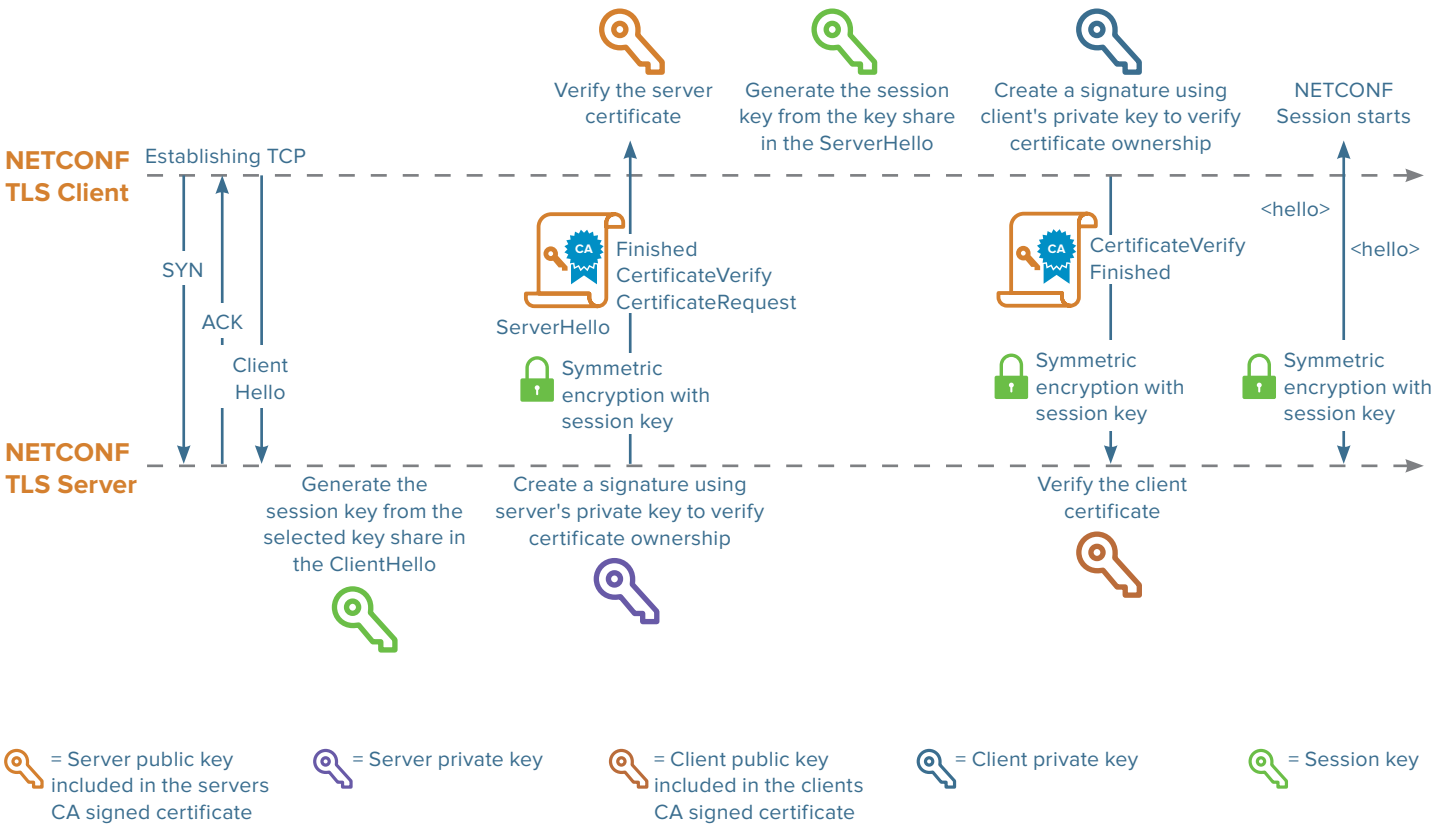
Imagine that an attacker somehow obtains a private key. They can now impersonate the identity associated with that key, because they can successfully decrypt NETCONF or RESTCONF messages that were encrypted using the public key embedded in any corresponding certificates. To prevent this, you need a way of invalidating the certificate immediately rather than waiting for its expiry date. This invalidation is called “certificate revocation” and can be achieved by maintaining a Certificate Revocation List (CRL) of certificates that should no longer be accepted.

A CRL is a time-stamped list identifying revoked X.509 certificates that are signed by a CA or CRL issuer and made freely available in a public repository. A revoked certificate is identified in a CRL by its certificate serial number. When the NETCONF or RESTCONF client or server receives a certificate, the client or server typically also acquires the most recently issued CRL and checks that the certificate serial number is not on that CRL. An advantage of this revocation method is that CRLs may be distributed via untrusted servers and untrusted communications.

4. X.509 Certificates for Transport Layer Security (TLS) Mutual Authentication

NETCONF and RESTCONF client-server applications can use the TLS v1.3 protocol [RFC 8446] to transmit encrypted data across a network in a way designed to prevent eavesdropping and tampering [RFC7589]. TLS provides NETCONF and RESTCONF and with a secure channel over TCP and relies on X.509 certificates to set up the secure communication channels.

On opening a TLS connection, as part of the TLS handshake protocol, the NETCONF or RESTCONF client that initiates the connection receives an X.509 certificate from the server which it can check to make sure that it is talking to the server entity that it intended to reach. When sending an X.509 certificate to the client, the server also requests the client's certificate. The client authenticates the server, sends the server the client's certificate to authenticate the client, and then the NETCONF or RESTCONF session starts.



Mappings of Certificates to NETCONF and RESTCONF Usernames for TLS

To derive the client identity, i.e. the username, NETCONF and RESTCONF servers both use the mechanisms described in the NETCONF over TLS section 7 “Client Identity” of RFC 7589. At the time of the writing of this application note, the IETF work on NETCONF, RESTCONF, and TLS client and server YANG data models for the authenticated client identity (among other things) is in draft status in the IETF NETCONF working group and thus not yet supported by ConfD. Therefore, to enable ConfD to derive the client identity for the X.509 certificates, i.e. the username, NETCONF over TLS servers need to implement their own mechanism. That mechanism could, for example, be based on the current version for the NETCONF and TLS client and server IETF draft YANG data models or a simplified mechanism as implemented by the demo below.

NETCONF over TLS Demo

A demo that setups the GnuTLS library as the TLS server integrated with ConfD’s NETCONF server can be found in ConfD Developer on GitHub:
https://github.com/ConfD-Developer/ConfD-Demos/blob/master/x509tls-netconf

The demo uses the gnutls-cli client program to set up a TLS connection to the NETCONF TLS server.

The ConfD NETCONF TLS server is set up with mutual x.509 authentication according to RFC 7589 with two simplifications for demo purposes:

Demo Simplification 1

In this simple demo, because we are setting up our own system with a NETCONF client and server and the system is under our private control, it doesn’t matter whether anyone else trusts the NETCONF client or server — the important thing is that the NETCONF client and server can trust each other. Because the demo is a private system, we don’t need to use publicly trusted CAs. We can simply set up our own root certificate authority with a self-signed certificate.

Demo Simplification 2

We use the simplest possible way of deriving the NETCONF username from the certificate in the demo by assuming that only the “admin” username holds a valid client certificate. Real implementations will need to derive the client’s username as described above in the section "Mappings of Certificates to NETCONF and RESTCONF Usernames for TLS".

RESTCONF over TLS Demo

A demo that setups the ConfD webserver as the TLS server integrated with ConfD’s RESTCONF server can be found in ConfD Developer on GitHub:
https://github.com/ConfD-Developer/ConfD-Demos/blob/master/x509tls-restconf

The demo uses curl as the client program to set up a TLS connection to the RESTCONF TLS server.

The ConfD RESTCONF web/TLS server is set up with mutual X.509 authentication according to the RESTCONF RFC 8040 with two simplifications for demo purposes:

Demo Simplification 1
See simplification 1 for the NETCONF over TLS demo above. A self-signed certificate is used for this RESTCONF demo, too.

Demo Simplification 2
We simply provide a username and password that is used for establishing the client identity to the RESTCONF server. The effect is that the client is authenticated using both basic authentication and X.509 certificate authentication. The basic authentication is only necessary to map the username to the certificate. Token authentication is an alternate way of mapping the username to the certificate. Real implementations will need to derive the client's username as described above in the section "Mappings of Certificates to NETCONF and RESTCONF Usernames for TLS".

5. X.509 Certificates for Secure Shell (SSH) Mutual Authentication

NETCONF support for X.509 Certificates for use with SSH as specified by RFC 6187 is included in the IETF NETCONF working group NETCONF and SSH clients and servers YANG data models that at the time of the writing of this application note is in draft status.

NETCONF over SSH [RFC 6242] points to the SSH transport layer [RFC 4253] where the server is authenticated and also points to the user authentication protocol described in RFC 4252 that requires the support of a digital signature to authenticate the client to the server ("public key" authentication). While optional, "password" authentication is often supported too.

For digital signature validity, the authentication depends upon the strength of the linkage between the public signing key and the identity of the signer. But without mechanisms for ensuring integrity and authenticity, a relying party is vulnerable to masquerading attacks through public key substitution.

Digital certificates, such as those in X.509 certificates [RFC 5280] are used in many corporate and government environments to provide identity management. They use a chain of signatures by a trusted root certification authority and its intermediate certificate authorities to bind a given public signing key to a given digital identity.

Mappings of X.509 Certificates to NETCONF Usernames for SSH

At the time of the writing of this application note, the IETF is working on NETCONF and SSH client and server YANG data models. Among other things, mapping X.509 certificates to an authenticated client identity are in draft status by the IETF NETCONF working group and thus not yet supported by ConfD. Therefore, to enable ConfD to derive the client identity for a X.509 certificate, i.e. the username, NETCONF over SSH servers need to implement their own mechanism. That mechanism could, for example, be based on the current version for the NETCONF and SSH client and server IETF draft YANG data models or a simplified mechanism as implemented by the demo below.

NETCONF over SSH Demo

To illustrate a setup of a ConfD enabled NETCONF SSH server that uses X.509 certificates for mutual authentication according to RFC 6187, a simple demo of a client and server can be found in ConfD Developer on GitHub:
https://github.com/ConfD-Developer/ConfD-Demos/edit/master/x509ssh-netconf

The demo uses the PKIX-SSH fork of OpenSSH to enable X.509 certificates for SSH. The PKIX-SSH client is used to send and receive NETCONF requests/replies, while the PKIX-SSH server is integrated with ConfD's NETCONF server as an external SSH server.

Demo Simplification 1

See simplification 1 for the NETCONF over TLS demo above. A self-signed certificate is used for this NETCONF over SSH demo as well.

Demo Simplification 2

We use the simplest possible way of deriving the NETCONF username from the certificate in the demo by assuming that only the "admin" username holds a valid client certificate. Real implementations will need to derive the client's username as described above.

6. Security Considerations

Requiring an X.509 certificate for authentication and encryption and a certificate revocation list will minimize the risk of for example man-in-the-middle attacks. It also requires the NETCONF or RESTCONF server or client to follow the IETF RFC guidelines for NETCONF or RESTCONF over TLS or SSH and always check if the certificate is self-signed or if the certificate is signed by a trusted CA and if the certificate has been revoked.

Avoid sharing certificates across NETCONF or RESTCONF clients and servers. It may seem like a management burden to set up individual identities and certificates for each server and client, but it means you can revoke the certificate for one client or server without having to reissue a new one to all your clients and servers. It also allows for a separation of concerns for your servers whereby each identity that is associated with a client's certificate can be granted a separate set of permissions through NACM.

7. For More Information

For more information about ConfD, visit *https://www.tail-f.com*

tail-f a Cisco company

www.tail-f.com
info@tail-f.com

Corporate Headquarters

Sveavagen 25
111 34 Stockholm
Sweden
+46 8 21 37 40