NETCONF and RESTCONF
Client/Server Models

Drafts covered:
• draft-ietf-netconf-system-keychain-00
• draft-ietf-netconf-ssh-client-server-00
• draft-ietf-netconf-tls-client-server-00
• draft-ietf-netconf-netconf-client-server-00
• draft-ietf-netconf-restconf-client-server-00

NETCONF WG
IETF 96 (Berlin)
Recap

• The IETF 95 (Buenos Aires) meeting led to a decision to split the “server-model” draft into a set of drafts more suitable to the models defined within them.

• The May 18 interim led to the final decision on how to do the split (see next slide).
The Selected Split Strategy (Proposal #3)

YANG Modules:

- ietf-system-keychain
- ietf-ssh-client
- ietf-ssh-server
- ietf-tls-client
- ietf-tls-server
- ietf-netconf-client
- ietf-netconf-server
- ietf-restconf-client
- ietf-restconf-server

How YANG modules map to drafts:

- draft-ietf-netconf-system-keychain
- draft-ietf-netconf-ssh-client-server
- draft-ietf-netconf-tls-client-server
- draft-ietf-netconf-netconf-client-server
- draft-ietf-netconf-restconf-client-server
Patterns

ietf-ssh-client == ietf-tls-client
ietf-ssh-server == ietf-tls-server
ietf-netconf-client == ietf-restconf-client
ietf-netconf-server == ietf-restconf-server

Key takeaway: The solutions/issues/resolutions in one almost always apply to the other...
Primary Update:
- Added “user-auth-credentials” to ietf-system-keychain
  - to support SSH-based clients
  - potentially useful to other types of clients (HTTP, SMTP, etc.)

```
+--rw user-auth-credentials
  +--rw user-auth-credential* [username]
    +--rw username string
    +--rw auth-method* [priority]
    +--rw priority uint8
    +--rw (auth-type)?
      +--:(certificate)
        | +--rw certificate* -> /keychain/private-keys/private-key/certificate-chains/certificate-chain/name
        +--:(public-key)
        | +--rw public-key* -> /keychain/private-keys/private-key/name
        +--:(ciphertext-password)
        | +--rw ciphertext-password? string
        +--:(cleartext-password)
        +--rw cleartext-password? string
```

The idea is that a client may login using a username + “some credentials”
- a client certificate
- a public key (e.g. ssh-rsa)
- a password

Idea is that a cleartext-password is configured, which gets converted by the server into a ciphertext-password.
Open Issues:

#1: How to encode certificate chain?
#2: How to handle semi-configurable aspects of keychain?
#3: Should we rename keychain the keystore?
#4: Does 'user-auth-credentials' have correct scope?
#5: YANG annotation for cleartext-password --> ciphertext-password?
#1: How to encode certificate chain?
(related to zerotouch #16)

• Original idea was that it would be easy on clients if a chain was modeled as a leaf-list of X.509v3 certificates, but:
  – having a list called “certificate-chain” is clunky
  – and there are formats designed to encode a chain...

• Options:
  1. keep what we have
  2. use a PKCS#12 structure from RFC 7292
  3. use a PEM “file” containing multiple BEGIN/END tags
  4. allow either PKCS#12 or PEM to be configured

OpenSSL can translate between PKCS#12 and PEM well enough.

Option #1 (what we currently have)

  +--rw certificate-chains
     |   +--rw certificate-chain* [name]
     |     |   +--rw name string // leafref target
     |     |   +--rw certificate* binary // an X.509v3

Option #2 (use a PKCS#12)

  +--rw certificates
     |   +--rw certificate* [name]
     |     |   +--rw name string // leafref target
     |   |   +--rw pkcs12 binary // a PKCS #12
#2: How to address the semi-configurable aspects of the keychain model?

- This issue is still open.
  - Last email to list sent last week, but no conclusion yet.

- To be honest, this is something that likely can only be discussed on list, as a concrete proposal with examples is needed to close it.
#3: should we rename keychain the keystore?

- Originally we had “ietf-keychain”
  - modelled after Mac OS X’s “Keychain Access” utility

- We renamed to “ietf-system-keychain” in current draft to disambiguate from the routing area’s “ietf-key-chain” module.
  - Regardless, it’s still causing confusion...

- Another common term for such a mechanism is “keystore”
  - so renaming to keystore would also makes sense.

- Do we just rename the module, or the draft name as well?
#4: Does 'user-auth-credentials' have correct scope?

Should the authentication credentials be limited to SSH?
   – do we want to support, for example, an HTTP client logging into an HTTP server?

If truly just for SSH, maybe the node should be renamed to something like ‘ssh-user-auth-credentials’?

Let’s wait and see on this one...
#5: YANG annotation for cleartext-password --> ciphertext-password?

- Current YANG module has:
  
  ```
  +--:(ciphertext-password)
  | +--rw ciphertext-password? string
  +--:(cleartext-password)
       +--rw cleartext-password? string
  ```

- The idea is that a cleartext-password is configured, which gets converted by the server into a ciphertext-password, at which point the cleartext-password is deleted by the server.

- But shouldn't there be a YANG annotation to describe this behavior to clients? – otherwise they’ll forever think that they are out of synch!
Updates:
  – added a note that '0.0.0.0' and '::' might have special meanings for the "inet:ip-address" type
    • should this information go into an update of RFC 6021?
  – added ‘ietf-ssh-client’ module (thanks to Gary Wu)
    • Next slide has details

Open Issues:
  None
The ietf-ssh-client Module

**initiating-ssh-client-grouping**

Assumption is that the application will handle opening the TCP connection

```
+----- server-auth
|   +----- trusted-ssh-host-keys?  -> /kc:keychain/trusted-ssh-host-keys/name
|   +----- trusted-ca-certs?       -> /kc:keychain/trusted-certificates/name {ssh-x509-certs}?
|   +----- trusted-server-certs?   -> /kc:keychain/trusted-certificates/name
+----- client-auth
    +----- matches* [name]
        +----- name?          string
        +----- match* [name]
            |   +----- name?       string
            |   +----- trusted-ssh-host-keys?  -> /kc:keychain/trusted-ssh-host-keys/name
            |   +----- trusted-ca-certs?       -> /kc:keychain/trusted-certificates/name
            |   +----- trusted-server-certs?   -> /kc:keychain/trusted-certificates/name
    +----- user-auth-credentials?  -> /kc:keychain/user-auth-credentials/user-auth-credential/username
```

**listening-ssh-client-grouping**

Same as above, but also configures which port the server opens to listen for connections on.

```
+----- address?    inet:ip-address
+----- port?       inet:port-number
+----- <uses ‘initiating-ssh-client-grouping’ above>
```

Like an access control list:
- if server==<name>
- then use specified credentials

There is a bug in here. The leafrefs should point to private-keys instead.
draft-ietf-netconf-tls-client-server

Updates:

- Same note regarding '0.0.0.0' and '::'
- Ran out of time to add a “ietf-tls-client” module

Open Issues:

None
Updates:
- Added in new features 'listen' and 'call-home' so future transports can be augmented in
- added ‘ietf-netconf-client’ module (thanks to Gary Wu)
  • Next slide has details

Open Issues:
#1: ssh-listen feature? (discussed on slide after next)
The ietf-netconf-client Module

```text
++-rw netconf-client
    +-rw initiate {initiate}?
    |    +-rw netconf-server* [name]
    |    |    +-rw name string
    |    |    +-rw (transport)
    |    |    |    +-:(ssh) {ssh-initiate}?
    |    |    |    |    +-rw ssh
    |    |    |    |    |    +-rw address inet:host
    |    |    |    |    |    +-rw port? inet:port-number
    |    |    |    |    |    |    +- <uses ‘initiating-ssh-client-grouping’ from the ietf-ssh-client module>
    |    |    |    |    |    +- missing TLS support here!
    +-rw listen {listen}?
        +-rw max-sessions? uint16
        +-rw idle-timeout? uint16
        +-rw endpoint* [name]
        |    +-rw name string
        |    +-rw (transport)
        |    |    +-:(ssh) {ssh-listen}?
        |    |    |    +-rw ssh
        |    |    |    |    +- <uses ‘listening-ssh-client-grouping’ from the ietf-ssh-client module>
        |    |    |    +- missing TLS support here!
```

This is for configuring a device to initiate persistent/periodic NETCONF connections. Not NETCONF Call Home (that’s in the server model).

This is for configuring a device to listen for NETCONF Call Home connection. Not normal port 830 connections (that’s in the server model).
Issue #1: ssh-listen feature?

• This issue only regards the ietf-netconf-server module
  – i.e. not the client module.

• Question received:

  Why is there an ssh-listen feature in the ietf-netconf-server.yang? SSH is a mandatory protocol for Netconf, so why make it optional?

• Thoughts:
  – NETCONF requiring the SSH transport is one thing, whether the server listens or initiates NETCONF Call Home connections is another. Call Home came later.
  – Some device may only want to support call home (e.g. BBF)
draft-ietf-netconf-restconf-client-server

Updates:
  – Same new features ('listen' and 'call-home') added
  – Ran out of time to add a “ietf-tls-client” module

Open Issues:
  – None
Next Steps

• Add missing TLS and RESTCONF client models
  – please reach out if you’d like to help!

• Close open issues

• Update the NETCONF Call Home reference implementation
  – **Warning:** just netconf-ssh and netconf-ch-ssh
    • No TLS or RESTCONF (is this a problem?)

• We will for sure be discussing this draft again at IETF 97!