Reverse SSH

Motivation

- Initial Device Discovery (using NetConf)
  - A "call home“ mechanism similar to TR069
- On-going Management (using NetConf)
  - The device may be deployed behind a NAT-ing device that doesn't provision an external address or port to connect to.
  - The device may be deployed behind a firewall that doesn't allow SSH access to the internal network.
  - The device may be configured in "stealth mode" with no open ports
  - The device may access the network in a way that dynamically assigns it an IP address and is not configured to use a service to register its dynamically-assigned IP address to a well-known domain name.
  - The operator prefers to have one open-port to secure in the data center, rather than have an open port on each device in the network.

History

- 2005
  - ?
- Early May
  - Submitted draft-kwatsen-reverse-ssh-00
  - Discussed on SAAG and IETF-SSH mailing lists
- Early June
  - Submitted draft-kwatsen-reverse-ssh-01
  - More discussion leading to two more solutions
- Early July
  - Requested guidance on NetConf and OPSAWG mailing lists

Fundamental “First-Time” Issues

- How does device identify itself
- How does device authenticate itself to the application
- How does application authenticate itself to the device

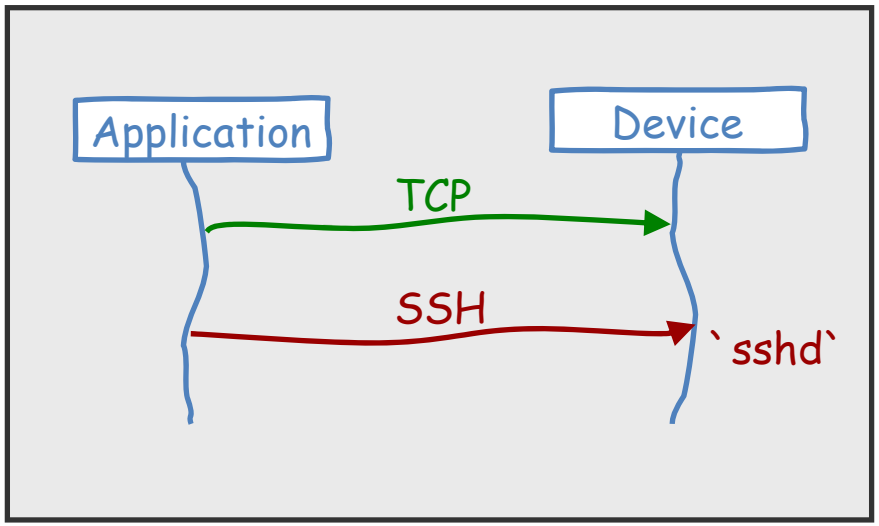
Four Proposals

- Discussions with SAAG and IETF-SSH have reached their logical conclusion, with four possible solutions:

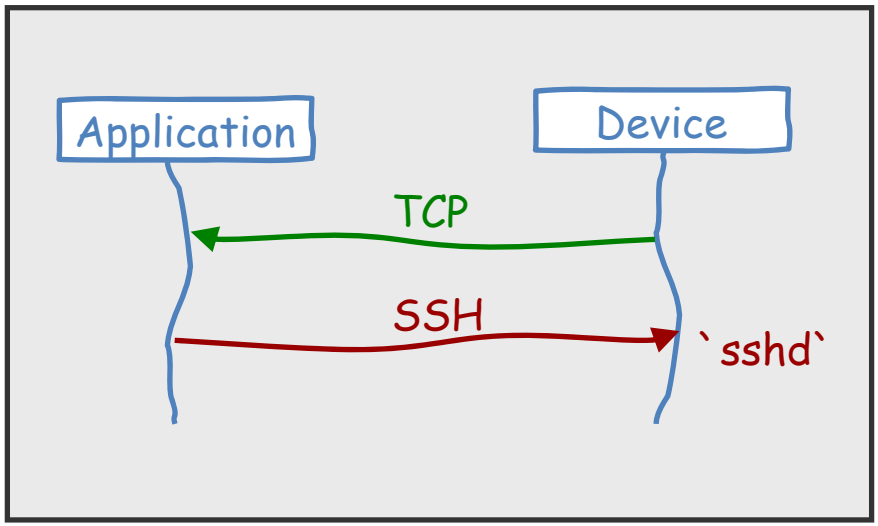
1. Out-of-band role-reversal
2. Out-of-band reversal, but no special ssh-role-reversal message
3. In-band solution, by allowing the SSH server to open channels
4. In-band solution, based on remote port-forwarding

#1. Out-of-band role-reversal

- Regardless of which peer initiates the underlying TCP connection:
  - The device is always the SSH-server
  - The application is always the SSH-client



Application-initiated connection



Device-initiated connection

#2. Out-of-band reversal, but no special ssh-role-reversal message

- Similar to option #1, but replaces special bootstrap negotiation with inline negotiation
  - SSH-KEYINIT message negotiates the MAC and Host-Key Algs to use
  - In order for Only Host-Key Algs allowed are x.509-* family and newly proposed HMAC-* family algorithms

#3 In-band solution, by allowing the SSH server to open channels

- Device SSHs to Application (SSH Server), which then opens channels back to the SSH client
- Device's login to application triggers it to differentiate it from standard SSH clients
- Assume relaxation in RFC stating the clients SHOULD reject server-initiated channel-open requests

#4 In-band solution, based on remote port-forwarding

- Device SSHs to Application (SSH Server), which then SSHs back to Device over SSH tunnel
- Device sets up a remotely forwarded-port
- Device's login to application triggers it to differentiate it from standard SSH clients

PROs/CONs

- 1st proposal is trivial to implement, but isn't a "pure" protocol solution, and so is unappealing to IETF-SSH list members
- 2nd proposal is not too hard to implement, but requires support for the x.509-* and HMAC-* Host-Key algorithms. Ultimately adds more complexity for little value
- 3rd proposal is trivial RFC change (s/SHOULD/MAY/), but even list members admit that it's unlikely that upstream OpenSSH would ever support it.
- 4th proposal requires no RFC and may be implemented today (?) – the implementation would necessitate bidirectional client/server authentication and double-encryption and

Questions

- Now that we've exhausted options with SAAG and IETF-SSH lists, should we:
  - Drop for a few more years
  - Proceed with one of their favored proposals (3 or 4)
  - Lobby more for one of the early solutions (1 or 2)
  - Push one of the earlier solutions in another WG (OPSAWG?) and just ask for them to review it?
  - Any other ideas?