Autonomic and Secure Boostraping of Network Devices

Robin Jäger
Hochschule Darmstadt
Controlware GmbH
Waldstraße 92, 63128 Dietzenbach, Germany
Email: robin.jaeger@controlware.de

***Abstract*—The networking industry is lacking a secure and truly universal zero touch protocol for network devices. Proprietary solutions based on mechanisms like DHCP exist, but do not offer the desired functionality, especially with regard to security. As each manufacturer develops a solution for its own ecosystem, the solutions are not compatible with each other.**

This paper presents a solution for this problem, using the management protocol NETCONF to transmit bootstrapping informations. To ensure security the device uses its 802.1AR DevID certificate for authentication while trust is achieved by an ownership voucher. This is a cryptographic artifact signed by the manufacturer to assure that the bootstrapping server belongs to the legitimate owner of a device.

I. Introduction

The current state of the art of most proprietary zero touch provisioning protocols is to use a common protocols like DHCP or DNS to discover a device and transfer information to it. This has several limitations. Among others the three most significant are:

1) no authentication
2) potential confidential informations transmitted in clear text
3) limited message size

As stated in [2] "it is commonly accepted that the initial connections between nodes are insecure". So the problem is not to use DHCP for general discovery. Rather, the problem is to use DHCP for both discovery and information transfer without authenticating the device.

Furthermore there is the lack of an universal approach for zero touch provisioning. As described most proprietary approaches depend more or less on the same protocols but are incompatible with each other. That is partly because vendors can use e.g. DHCP options as it suits their individual implementation. More importantly most operations are based on outdated CLI commands and flat configurations files. These incompatibilities are a huge challenge for vendor independent system integrators and institutions operating a multi vendor network.

To sum up, a solution must meet two criteria:

- secure and mutual authentication
- universal information transfer without flat configuration files

II. A secure approach for device bootstrapping

This paper describes an approach inspired by "opportunistic security" ([3]) and tries to achieve maximum security whenever it is possible. As a consequence the first steps of the the discovery phase are more or less unsecured, because the devices did not exchange any information and no trust has been established. Each of the following steps is then as well secured as the information level of the communication partners permits.

A. Discovery

If an integration in an existing infrastructure is intended, it is possible to use a common protocol like DHCP for discovery. However, as [5] states, there are still some efforts to be made in order to standardize e.g. the use of DHCP options.

A alternative procedure for the discovery uses GRASP [6]. Since this is an passive approach for the device, this should be the preferred option. It is also the protocol used in the proof of concept for this paper (see chapter III).

In this case, the pledge waits on the GRASP M_Flood of an registrar after completing the boot sequence. If the bootstrap is done in an remote location, the pledge may also wait for the flood message of a proxy as described in [2]. For maximum flexibility the current implementation listens for both messages, but prefers the direct connection to the registrar, it its available. If the bootstrap is performed in a local network, no proxy is needed and the devices can directly communicate with the registrar.

Only if no M_Flood was received after a certain period of time the pledge should try other discovery methods like DHCP or browse other sources of bootstrapping data like removable storage.

B. Authentication

A new device should not get access to the network right away. There are two ways to separate the device from the network:

1) certificate based access control
2) virtually separating the device from the network

Both approaches would use an authentication very similar as described in [2]. The pledge generates itself a IPv6 link-local address and sends an authentication request to the registrar. This is done using an RESTful HTTP Message to an (yet to

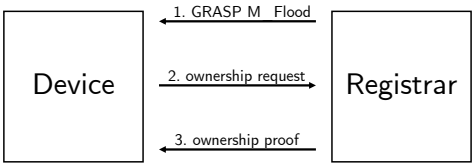

Fig. 1. Discovery and Authentication

be) standardized URI. In this request the pledge authenticates itself using its IDevID certificate. The registrar returns an ownership voucher as described in [1] and [2]. Basically the voucher is a cryptographic secure file signed by the vendor. Because the pledge hast a datastore with all trusted certificates, the pledge can easily verify if the signature of the ownership voucher. Figure 1 shows the simplified process of discovery and authentication.

If necessary the pledge receives a domain certificate to connect to the network. This is done by EST as described in [RFC 7030]. At the time of writing this is the preferred way, because it is technically feasible and more universal than a virtual network separation.

Concepts to accomplish a virtual separation like VLANs, MPLS or VxLAN are highly depended on the underlaying network architecture. Approaches like network slicing (e.g. [4]) might be interesting, but there is no available implementation yet. Furthermore, it depends strongly on the deployment scenario whether a fully grown software defined network slicing approach as proposed for 5G [4] might be overkill for a problem like this.

C. Bootstrap

After mutual authentication the actual bootstrap process begins. The device opens a NETCONF server on the standard ports and waits for the bootstrap data. The first message in this process is once again a ownership voucher send by the provision server in a dedicated RPC. This is necessary because registrar and server are not necessarily the same system. After validating this voucher the device stores the certificate to verify the bootstrap data later.

To convey the bootstrapping data, a separate RPC is used. The tree view of the used YANG Module is described in Appendix A. The main informations needed for the bootstrapping process are:

1) boot image
2) pre configuration script
3) configuration
4) post configuration script

Additionally the module contains information how to download or validate said resources or how to interpret a script.

III. Implementation

A first prototype for the approach presented in this paper can be found on github[1]. The code is a proof of concept and is still under development. But it shows that the solution described here is technically possible. Especially the client side implementation is highly specific for every device and therefore the responsibility of the manufacturer.

IV. Conclusion

Section I states security and universality as the two main requirements for a zero touch solution. The main challenge to fulfill security requirements is the mutual authentication of both peers. In order to achieve this, the device authenticates itself using a 802.1AR DevID certificate. Authentication of the bootstrap server is done with an ownership voucher which is signed by the manufacturer.

Universality is achieved by the use of the widespread management protocol NETCONF. The protocol can be extended by YANG modules and is therefore usable for autoprovisioning applications. These YANG modules must be standardized to ensure interoperability.

This also requires additional effort for the manufacturers. It will depend if NETCONF reaches a big enough popularity and whether the demand for for a truly universal solution will be large enough to force the manufacturers adhere the standards. Irrespective of these rather economic aspects, this paper and the associated proof of concept show that such a solution is technically feasible.

Acknowledgment

This paper is related to [5]. While this work focuses more on a practical implementation of a solution, the authors of [5] are working on a universal standardization of a similar protocol.

Also the author would like to thank Brian E Carpenter for his implementation of GRASP[2] and Christian Hopps for his NETCONF[3] implementation. Both projects provided a good starting point for further implementations.

References

[1] Watsen, K and Richardson, M and Pritikin, M and Eckert, T *A Voucher Artifact for Bootstrapping Protocols https://tools.ietf.org/html/rfc8366 2018*

[2] Pritikin, M and Richardson, M and Behringer, M and Bjarnason, S and Watsen, K , *Bootstrapping Remote Secure Key Infrastructures (BRSKI)* IETF Internet draft - 2019 - https://tools.ietf.org/html/draft-ietf-anima-bootstrapping-keyinfra-19

[3] Dukhovni, Viktor, *Opportunistic security: Some protection most of the time*, 2014

[4] Homma, S and Miyasaka, T and Galis, A and Ram OV, V and Lopez, D and Contreras-Murillo L. and Ordonez-Lucena, J and Martinez-Julia, P and Qiang, L and Rokui, R and Ciavaglia, L and de Foy, X *Network Slice Provision Models https://www.ietf.org/id/draft-homma-slice-provision-models-00.txt 2019*

[5] Watsen, Kent and Abrahamsson, Mikael and Farrer, Ian, *Zero Touch Provisioning for Networking Devices* IEEE, 2019 https://www.ietf.org/id/draft-ietf-netconf-zerotouch-29.txt

[1] see https://github.com/trjaeger/sauZTP
[2] see https://github.com/becarpenter/graspy
[3] see https://github.com/choppsv1/netconf

[6] Bormann, C and Carpenter, B and Liu, B and Jiang, S and Du, Z and Carpenter, B and Sun, Q, *A generic autonomic signaling protocol (grasp)*, 2017 - IETF Internet Draft - https://tools.ietf.org/html/draft-ietf-anima-grasp-15

APPENDIX A
RELEVANT YANG MODULES

For simplicity and due the lack of standardized modules, most values are defined as strings, although in the future more specific types for e.g. the hash algorithm will be suitable.

```
module: bootstrap-information
+--rw bootstrap-information
   +--rw id                          string
   +--rw boot-image
   |  +--rw name                     string
   |  +--rw version                  string
   |  +--rw download-uri             string
   |  +--rw verification
   |     +--rw hash-algorithm?       string
   |     +--rw hash-value?           string
   +--rw configuration-handling?     string
   +--rw pre-configuration-script
   |  +--rw filename                 string
   |  +--rw interpreter              string
   |  +--rw download-uri             string
   |  +--rw verification
   |     +--rw hash-algorithm?       string
   |     +--rw hash-value?           string
   +--rw configuration?              binary
   +--rw post-configuration-script
      +--rw filename                 string
      +--rw interpreter              string
      +--rw download-uri             string
      +--rw verification
         +--rw hash-algorithm?       string
         +--rw hash-value?           string
```