July 2022

ENFORCING QUALIFIED IMAGE AND ANTI-ROLLBACK PROTECTION ON A DEVICE THROUGH OWNERSHIP VOUCHER

Jabir Mohammed

Reda Haddad

Bazil Mohammed Ali

Srihari Raghavan

ENFORCING QUALIFIED IMAGE AND ANTI-ROLLBACK PROTECTION ON A DEVICE THROUGH OWNERSHIP VOUCHER

AUTHORS:
Jabir Mohammed
Reda Haddad
Bazil Mohammed Ali
Srihari Raghavan

ABSTRACT

Techniques are presented herein that extend an ownership voucher (OV) to carry software artifact hashes. The presence of such hashes supports a number of functionalities, including enforcing the loading of a specific qualified image to given hardware along with rollback prevention; preventing the installation of other released images through a Universal Serial Bus (USB), an installation workflow, or a network; avoiding pre-staging errors as well as administrative errors on loading images; conveying the known good values of software artifacts for a given release to a customer for a specified release; and aiding operational simplicity and automation using Manufacturer Authorized Signing Authority (MASA) workflows to facilitate easy deployments without manual intervention.

DETAILED DESCRIPTION

Generally, a software vendor may release-sign and deliver various images, which may then be downloaded through an online distribution facility. Since all of those images are release-signed, any image may be booted in a customer's hardware. However, a customer may have a special requirement to qualify a specific release and the customer may want to enforce the loading of only a qualified image in their hardware. Such enforcing may also prevent the loading of images other than the qualified image at later points in time using an upgrade or downgrade installation workflow, a Universal Serial Bus (USB) workflow, as well as through network booting. Additionally, through such an enforcement mechanism a software vendor could inform a customer about the current stable software artifacts for a given release.

At present, such an enforcement mechanism is not available. To address that lack, techniques are presented herein that extend an ownership voucher (OV, see the Internet

Engineering Task Force (IETF) Request for Comments (RFC) 8572) to carry additional software artifact information (e.g., software hashes). Such software artifact information may also help a software vendor or image provider to convey the known good values of software artifacts for a given release. Since an OV is part of a Manufacturer Authorized Signing Authority (MASA) workflow, such an approach may help with operational simplicity through automation.

Aspects of the techniques presented herein focus on providing hashes of certain software artifacts through an extended OV. By employing such techniques, a customer may, for example, decide on a qualified release and ask a MASA to issue an OV based on that specific release.

A MASA may then check the latest known hash from various software artifacts for that specific release and include the hashes in an extended OV, taking the known hashes from a release-signed repository. The described approach may be extended to include other software artifacts or other package management system hashes. At a minimum, an extended OV (according to the techniques presented herein) may include the following hashes:

- A basic input/output system (BIOS) known good hash for the specific hardware.
- A GNU GRand Unified Bootloader (GRUB) known good hash for the specific software.
- A GRUB configuration hash for the specific software.
- A kernel image hash for the specific software.
- An initial random-access memory (RAM) disk (`initrd`) known good hash for the specific software.
- Optionally, the latest database certificate combination hashes for the specific hardware.
- Optionally, the hashes of any critical software artifacts in the future.

As noted previously, aspects of the techniques presented herein leverage an OV. Among other things, an OV artifact may be used to securely identify a device's owner as it is known to the manufacturer, and it is signed by the device's manufacturer. The structure of an OV is defined in Section 3.3 of RFC 8572. In particular, the `ietf-voucher` artifact is defined in RFC 8366:

```
module: ietf-voucher
  yang-data voucher-artifact:
      +---- voucher
      +---- created-on                        yang:date-and-time
      +---- expires-on?                       yang:date-and-time
      +---- assertion                         enumeration
      +---- serial-number                     string
      +---- idevid-issuer?                    binary
      +---- pinned-domain-cert                binary
      +---- domain-cert-revocation-checks?    boolean
      +---- nonce?                            binary
      +---- last-renewal-date?                yang:date-and-time
```

Aspects of the techniques presented herein extend the structure of an OV. Elements of that extension encompass a new data structure, `known-hash-list`:

```
known-hash-list:: SEQUENCE {
Name             ;
HashType hashtype ;
Hash     hash_data;
};
```

Following the above-described extension, the `ietf-voucher` artifact becomes (with the new elements highlighted in bold):

```
module: ietf-voucher
  yang-data voucher-artifact:
      +---- voucher
      +---- created-on                        yang:date-and-time
      +---- expires-on?                       yang:date-and-time
      +---- assertion                         enumeration
      +---- serial-number                     string
      +---- idevid-issuer?                    binary
      +---- pinned-domain-cert                binary
      +---- domain-cert-revocation-checks?    boolean
      +---- nonce?                            binary
      +---- last-renewal-date?                yang:date-and-time
      +---- release-version                   string
      +---- known-hash-list                   known-hash-list
```

In the extended `ietf-voucher` artifact that was presented above, the `known-hash-list` is a list which carries the hashes of all of the artifacts. Each element of the list includes (as defined in the above-presented data structure) a Name field representing the name of the artifact, a HashType field that identifies a hash algorithm type (e.g., SHA-256, SHA-512, or SHA-3), and a Hash field that represents the known good values hash of the artifact.

An OV may be installed in a number of different ways, including through a command line interface or an application programming interface (API)-based workflow or through Secure Zero Touch Provisioning (SZTP, see RFC 8572 and RFC 8366).

Once an OV is installed using any of the above methods, when an image is downloaded it is possible to verify the various software artifact hashes against what is stored in the OV to enforce the booting of that image. Additionally, the stored OV may be used as rollback protection mechanism. Checking a software artifact hash against an OV will prevent a system from booting any non-qualified software images. As a result, on a given set of hardware one cannot change the image without reissuing a new voucher on to the hardware. During an upgrade or downgrade workflow, a USB workflow, or a network workflow, the same extended ownership hash value may be used to again enforce the loading of only a qualified image.

Consequently, if a customer wishes to load a new image the customer must reach out to a MASA with the device information and the new qualified release version and request that the MASA issue an OV for that specific release.

Figure 1, below, depicts elements of an exemplary sequence flow according to aspects of the techniques presented herein and reflective of the above discussion.

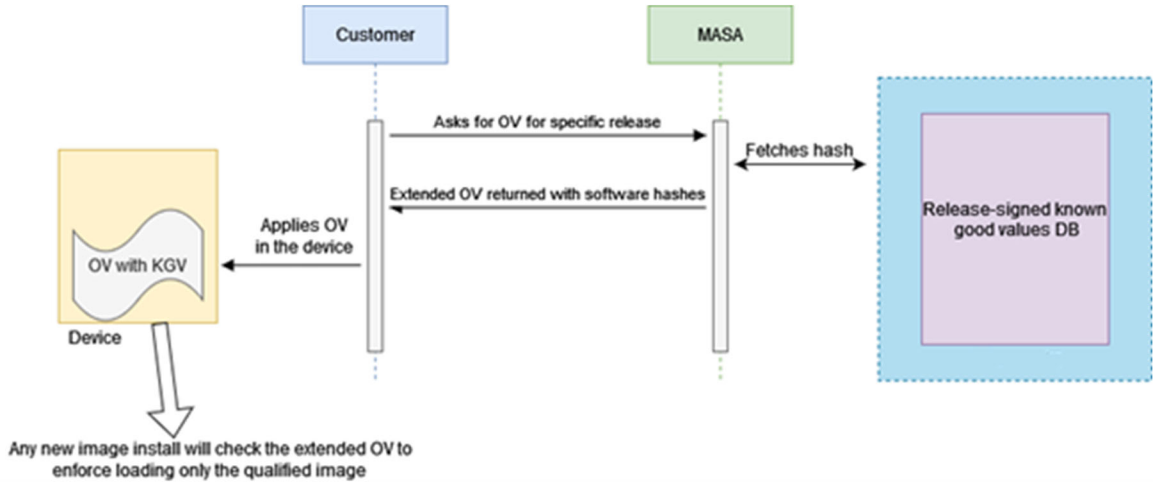

Figure 1: Exemplary Sequence Flow

According to aspects of the techniques presented herein, anti-rollback protection may be enforced through the cross-verification of the artifact hashes of a downloaded

image against the OV that was conveyed with the image artifact hashes. While an image may be downloaded through a USB workflow, through SZTP, through an installation workflow, or through a network iPXE, every artifact must be cross-verified against the issued extended voucher, which carries the known good value of various artifacts.

Figure 2, below, depicts elements of an exemplary process flow according to aspects of the techniques presented herein and reflective of the above discussion.

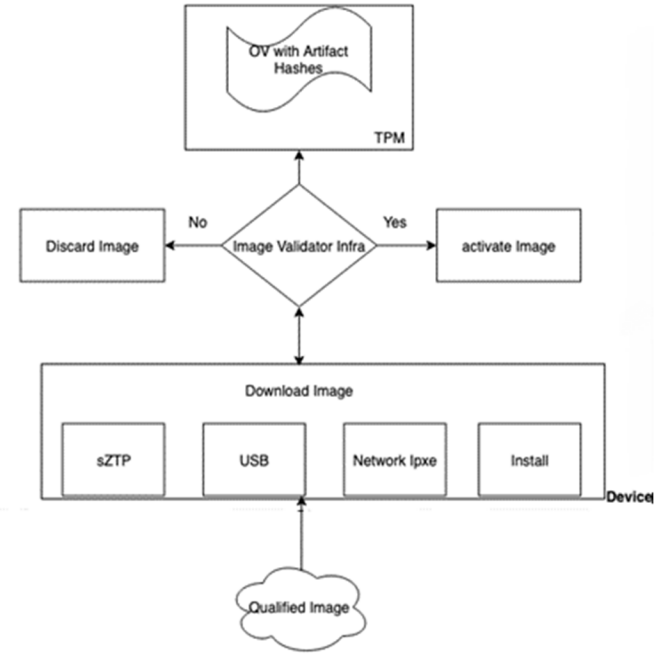

Figure 2: Exemplary Process Flow

During voucher generation, a customer may indicate the specific software release to which the OV should be tied. Based on the release selection, a manufacturer may embed the various artifact hashes by communicating with a release trusted store which holds the known good value per release. If the customer wishes to load a different image, then the customer must request a new OV from the manufacturer with that specific release. Through this process, an OV enforces rollback protection and guaranties that an image will run with only a qualified image.

In summary, techniques have been presented herein that extend an OV to carry software artifact hashes. The presence of such hashes supports a number of functionalities, including enforcing the loading of a specific qualified image to given hardware along with

rollback prevention; preventing the installation of other released images through a USB, an installation workflow, or a network; avoiding pre-staging errors as well as administrative errors on loading images; conveying the known good values of software artifacts for a given release to a customer for a specified release; and aiding operational simplicity and automation using MASA workflows to facilitate easy deployments without manual intervention.