Qualcomm

OpenAFS and Secure Boot

OpenAFS Workshop - June 10-11 2024

Erik Beatty | IT Cloud Architect, Sr Staff | Qualcomm Incorporated

What is Secure Boot?

- Part of the UEFI specification
- Tries to guarantee that only trusted software can be loaded
- Boot loader, OS, drivers, etc all need to be signed by a trusted certificate

The problem

- We use dkms to build and deploy the client OpenAFS kernel module
- As-is, the kernel module is unsigned and results in the following:
  - modprobe: ERROR: could not insert 'openafs': Operation not permitted (from yum output)
  - Lockdown: modprobe: unsigned module loading is restricted; see man kernel_lockdown.7 (from syslog)

Getting started

- Secure Boot allows for the import of a Machine Owner Key (MOK) that can be used to sign modules
- Install packages

```
$ yum install openssl mokutil keyutils
```

- Checking if Secure Boot is enabled

```
$ mokutil --sb-state
SecureBoot enabled
```

- Generate the cert and import
  - For some reason the MOK manager interface would not come up after reboot, I ended up having to use the ‘mokutil --set-verbosity true’

```
$ openssl req -new -x509 \
   -newkey rsa:2048 -keyout /root/openafs.key \
   -outform DER -out /root/openafs.der \
   -nodes -days 36500 -subj "/CN=OpenAFS Kmod Signing MOK"

$ mokutil --set-verbosity true

$ mokutil --import /root/openafs.der

$ reboot
```

Manually signing

- Next up, getting the dkms module to use the self-signed key
- For some reason dkms was not reading the SIGN_TOOL directive from the /etc/dkms/<module>.conf, I've read a few posts now that state that has been restricted to the /etc/dkms/framework.conf
- Just to do the initial validation I was able to unxz the /lib/modules/$(uname –r)/extra/openafs.ko.xz and manually run the script to sign it

  $ /usr/src/kernels/$(uname -r)/scripts/sign-file sha256 /root/openafs.key /root/openafs.der /lib/modules/$(uname –r)/extra/openafs.ko

- After that I re-xz'd the file and was able to insmod it successfully! So yes, we can self-sign and load!

Automating the signing

- Continuing on with trying to automate the process
- I noticed that the dkms build itself referenced the following

  Signing key: /var/lib/dkms/mok.key
  Public certificate (MOK): /var/lib/dkms/mok.pub

- Doing some digging and I found that the /etc/dkms/framework.conf has

  # mok_signing_key=/var/lib/dkms/mok.key
  # mok_certificate=/var/lib/dkms/mok.pub

- I dropped a custom.conf under /etc/dkms/framework.conf.d to set those values to the custom self-signed key

  mok_signing_key=/root/openafs.key
  mok_certificate=/root/openafs.der

- And that worked, the caveat being that it is used for all dkms modules, not just OpenAFS, but I suspect we would only deploy a single custom dkms key/cert anyways, so that should be fine.
- Doing some additional searches and I found that the /var/lib/dkms/mok.pub can be imported

  $ mokutil --set-verbosity true
  $ mokutil --import /var/lib/dkms/mok.pub
  $ reboot

- So I did that and voila! That worked as well!

Next steps

- Proof of concept wise, being able to self-sign an OpenAFS dkms built kernel module and load it in a Secure Boot environment is definitely doable
- The biggest hurdle is the import of the cert, having to reboot and enter the MOK manager on each host does not scale
- Investigate additional tools/utilities/flows
  - efi-updatevar – sounds very manual, but had the concept of delete all the keys and reload
  - certmule – appears to allow for the unattended enrollment of keys into the MOK, but requires a system owner key in the UEFI secure boot DB
  - redfish - use redfish to clear the keys, then the system is in "setup" mode for Secure Boot and then re-enroll all the keys starting with our own custom one as the Platform Key (PK) from the OS (in theory)
- Having discussions with various HW vendors
  - Can they support custom key enrollment before shipment?
  - How about key rotations?
  - Firmware updates?
  - Security policies to protect our keys?
- Obtain a “blessed” key to sign the module?

Links

- These were some helpful links discovered along the way
  - https://gist.github.com/lijikun/22be09ec9b178e745758a29c7a147cc9 – Example of signing dkms nvidia drivers in secure boot
  - https://github.com/dell/dkms#secure-boot – MOK overview as well as the import of the dkms mok.pub
  - https://blogs.oracle.com/linux/post/the-machine-keyring - certmule example
  - https://sysguides.com/fedora-uefi-secure-boot-with-custom-keys/ - lots of good details
  - https://redfishforum.com/thread/572/install-certificate-secure-boot - redfish examples
  - https://docplayer.net/151198843-Secureboot-certificate-management-by-using-redfish.html - more redfish examples

Thank you

Qualcomm

Follow us on:

For more information, visit us at:
qualcomm.com & qualcomm.com/blog