New issue

sign bootloader to make it compatible with UEFI secure boot? #42127

Open

KiaraGrouwstra opened on Jun 17, 2018

The install guide states:

> **UEFI boot**
> The EFI bootloader of the installation media is not signed and is not using a signed shim to boot. This means that Secure Boot will need to be disabled to boot.

I have a work machine I would like to use NixOS on.

Unfortunately, it uses secure boot, forcing me to choose between disabling secure boot (-> can boot from arbitrary USBs but can't use hard drive), or leaving it (can use hard drive but no NixOS).

It would be nice if NixOS were compatible with secure boot.

It seems this involves signing the bootloader with some Microsoft key. This makes me wonder, would this be possible, or would there be drawbacks to this?

(As a workaround, it seems one can also add a signing key to the UEFI firmware. However, if the NixOS bootloader is not signed yet I presume this does not apply yet.)

👍 111 🚀 2 👀 1

kalbasit on Jul 3, 2018 Member

Relevant Arch wiki: https://wiki.archlinux.org/index.php/Secure_Boot#Using_your_own_keys

👍 1

matthewbauer on Jul 29, 2018 Member

I have looked into signing in Nixpkgs (#38624) and basically came to the conclusion it is not a good idea right now. The problem is that we can't safely put private stuff in the Nix store (NixOS/nix#8). The private key will end up getting leaked into the .drv file which is available to anyone with access to your Nix store. When we start doing this on Hydra the key becomes pretty much worthless.

I think this is a pretty important thing to work towards, though. More and more computers are using secure boot.

👍 10

ElvishJerricco on Sep 10, 2018 Contributor

Would it be possible to sign files during activation, rather than during build? The problem, I suppose, would be that the activation script then depends on a signing key existing unencrypted on your file system, which is both impure and insecure. We can't have the activation script prompt the user for a decryption password, can we?

👍 3

knedlsepp on Nov 9, 2018 Member

Has anyone tried getting something working by using one of the signed shim bootloaders other distros are using?

etu on Nov 11, 2018 Contributor

@knedlsepp Hey, someone mentioning having it set up in the nixos IRC-channel the other day.

Here's the piece of log covering this: https://logs.nix.samueldr.com/nixos/2018-11-07#1541583001-1541583306;

This seems to be the relevant links:
https://git.joko.gr/nixos/gummibootr
https://git.joko.gr/joko/etc-nixos/src/master/machines/munch.nix#L10

I haven't tried it myself though.

redfish64 on Jan 6, 2019 · edited by redfish64 Edits Contributor

@etu I tried this and got it to work

Open **sign bootloader to make it compatible with UEFI secure boot?** #42127

I looked at the code, specifically https://git.joko.gr/nixos/gummibootr/src/master/systemd-boot-builder.py and it seems comprehensive. It take the kernel, initird, and kernel parameters and sticks them all into one efi binary, and signs that binary using keys hardcoded as "/etc/uefi/DB.crt" and "etc/uefi/DB.key". As for my unskilled appraisal, this looks satisfactory.

Neutral things:

It recreates and signs *all* generations of the kernel that have not been garbage collected (ie. seen in systemd-boot as the prompt with names somewhat like Gen 60, Gen 61, Gen 62, ...). This sort of makes sense because if the keys have changed since "Gen 60" has been created and now we are on "Gen 63", for example, you do not want to *not* be able to run "Gen 60" using secure boot, even though it was created before the keys were changed. Realistically, though, the keys aren't going to change very often, so I'm not sure if I like this, especially because it causes more problems, see below (although this can be fixed).

You'll need efitools to create key pairs for the various EFI keys. I created a pull request for it, here: #53489 but it's not in nixpkgs yet.

Bad things:

Each kernel is understandably pretty big, because it now contains the initrd along with it (along with the kernel parameters, although I hope for godsake that in your system they themselves don't add much to the size...). The problem here is that because there are multiple generations that by default aren't cleaned up, the disk space fills up quickly.

I couldn't figure out how to get it to sign the initial "systemd-boot" (which in turn verifies and calls the kernel generations). It needs NIXOS_INSTALL_BOOTLOADER to be set to invoke that code. I simply hacked the code to remove the check. This check is also in the original "nixos/modules/system/boot/loader/systemd-boot/systemd-boot-builder.py" so I don't think is a good fix for the master branch, because I'm not sure of the implications... alternatively you could just sign it manually, using:

```
$ sbsign --key /etc/uefi/DB.key --cert /etc/uefi/DB.crt systemd-bootx64.efi --output systemd-bootx64.efi.signed
$ mv systemd-bootx64.efi.signed systemd-bootx64.efi
```

Very bad things:

If the disk space fills up on your ESP (where the kernel generations live), running nixos-rebuild will destroy all of your kernel generations and not report an error, so if you aren't looking carefully at the output, next time you reboot, you won't be able to reboot into a working system using *any* of the generations because it tries to sign each one to another file, runs out of space and doesn't report the error (which, of course, happened to me*).

If I find the time I'll try and fix this.

* This can be fixed by disabling secureboot and using the nixos minimal boot disk, mount your encrypted root partition on /mnt and run:

```
$ nixos-enter
```

and clean up the old generations by running:

```
$ nix-env -p /nix/var/nix/profiles/system --delete-generations old
$ nix-collect-garbage -d --delete-older-than 1d
$ nixos-rebuild boot
```

ElvishJerricco on Jan 6, 2019 Contributor

We could potentially write an efi application that does extra verification on the grub.cfg and on the initrd, separately from the kernel itself. This way it can all be kept separate, and merely changing kernel params doesn't have to duplicate either the kernel or the initrd. Of course you then also have to make sure grub doesn't let users modify command line args before booting.

knedlsepp on Jan 6, 2019 · edited by knedlsepp Edits Member

I'm currently piggybacking on an Ubuntu shim +grub to boot nixOS. It works fine, but I couldn't figure out how to do this from within nixOS, so I simply installed nixOS on top of an existing Ubuntu.

ElvishJerricco on Jan 6, 2019 Contributor

You'd still need to write something in between shim and grub that actually verifies the relevant files with shim's protocol, right?

redfish64 on Jan 6, 2019 · edited by redfish64 Edits Contributor

> We could potentially write an efi application that does extra verification on the grub.cfg and on the initrd, separately from the kernel itself.

@ElvishJerricco This configuration doesn't use grub, but rather just systemd-bootx64.efi, which nixos fetches and builds using https://github.com/NixOS/systemd/ but otherwise your point stands. Since systemd is already forked, it wouldn't be too much work to do this.

OTOH, grub looks like it does this already, see https://ruderich.org/simon/notes/secure-boot-with-grub-and-signed-linux-and-initrd (although I haven't tried this), so it might be better just to alter the grub bootloader configuration in nixpkgs to support secureboot, rather than trying to hack systemd-boot.

lheckemann on Apr 26, 2019 Member

@grahamc apparently has a somewhat-working solution for this, although it has problems with running out of space on /boot?

grahamc on May 15, 2019 Member

See #53901

matthewbauer added this to the 19.09 milestone on May 27, 2019

stale on Jun 2, 2020

Thank you for your contributions.

This has been automatically marked as stale because it has had no activity for 180 days.

If this is still important to you, we ask that you leave a comment below. Your comment can be as simple as "still important to me". This lets people see that at least one person still cares about this. Someone will have to do this at most twice a year if there is no other activity.

Here are suggestions that might help resolve this more quickly:

1. Search for maintainers and people that previously touched the related code and @ mention them in a comment.
2. Ask on the NixOS Discourse.
3. Ask on the #nixos channel on irc.freenode.net.

stale added 2.status: stale on Jun 2, 2020

21 remaining items

Load more

Madouura on Nov 20, 2021 · Hidden as duplicate

Artturin on Nov 21, 2021 — Member

secure boot can be tested in a vm https://fedoraproject.org/wiki/Using_UEFI_with_QEMU?rd=Testing_secureboot_with_KVM#Creating_a_VM

👍 1

nixos-discourse on Jan 9, 2022

This issue has been mentioned on **NixOS Discourse**. There might be relevant details there:

https://discourse.nixos.org/t/encrypted-root-with-single-password-prompt/17054/3

ilikeavocadoes on Jun 3, 2022 — Contributor

Has there been any progress lately? Is there anything that can be done to help without having any know-how about UEFI?

👍 5

06kellyjac on Jun 3, 2022 — Member

There has been progress

NixOS/rfcs#125

https://github.com/DeterminateSystems/bootspec-secureboot

Im not sure how much more remains to be done other than finishing the RFC and stabilizing the bootspec format

👍 19 ❤️ 10

weitzj on Nov 12, 2022

Just a hint:
Would this project also be feasible to integrate with NixOs? Looks like it had secure boot available for ZFS

https://github.com/zbm-dev/zfsbootmenu

Atemu on Nov 12, 2022 — Member

Our bootloaders (systemd-boot and GRUB) already support secure boot. That's not the issue.

stehessel closed this as completed on Mar 12, 2023

stehessel reopened this on Mar 12, 2023

brainrake on Mar 13, 2023 · edited by brainrake — Edits — Contributor

secure boot support: https://github.com/nix-community/lanzaboote
pr: #202497

🎉 3 ❤️ 9 🚀 3

chuangzhu on Mar 18, 2023 · edited by chuangzhu — Edits — Contributor

> Just a hint: Would this project also be feasible to integrate with NixOs? Looks like it had secure boot available for ZFS
>
> https://github.com/zbm-dev/zfsbootmenu

I also want to mention ZFSBootMenu. It is a unique approach that decrypts the filesystem at the boot loader stage so you can put the kernel and initrd encrypted in / alongside with the system. In this approach you don't have to sign kernel and initrd every time they update. However there are some drawbacks:

- ZFSBootMenu doesn't have a file where you can specify the kernel cmdline for each boot entry (`init=` is needed for NixOS).
- ZFSBootMenu doesn't support TPM2. An attacker may just hook a CH341a to your BIOS ROM chip and turn off secure boot.

samueldr on Mar 18, 2023 — Member

ZFSBootMenu also uses `kexec`, which on x86_64 will most likely be fine, but is still YMMV.

nixos-discourse on Aug 20, 2023

This issue has been mentioned on **NixOS Discourse**. There might be relevant details there:

https://discourse.nixos.org/t/dual-boot-on-a-secure-boot-enabled-system/31963/1

Artturin mentioned this on Oct 7, 2023
Secure Boot support #255643

JohnRTitor mentioned this on Dec 6, 2024
Secureboot with Shim NixOS/org#45

Add a comment

Write | Preview

Use Markdown to format your comment

Paste, drop, or click to add files

Close issue | Comment

Remember, contributions to this repository should follow its contributing guidelines, security policy and code of conduct.

Metadata

Assignees
No one assigned

Labels
No labels

Type
No type

Projects
No projects

Milestone
No milestone

Relationships
None yet

Development
No branches or pull requests

Notifications Customize

Subscribe

You're not receiving notifications from this thread.

Participants

 +22