Passphrase-less reboots using kexec under NixOS

by Dr. Katja Flinzner, Wanja Hentze • 17 August 2026

Encrypted hard drives protect data, including in the event of theft. Even if entire racks are carried out of the server room: without the correct passphrase for disk encryption, the hard drives are of little use. So of course we encrypt them. So far, so obvious.

This security, however, comes at a price. Every instance of human intervention during a reboot costs time and increases the risk of errors: if the boot process is interrupted, whether due to distraction or any imaginable incident, the server gets stuck at the password prompt and fails to boot up again.

We sometimes find ourselves needing to reboot servers more frequently. While a lot of software can be updated or reconfigured on the fly, this option stops at the Linux kernel: eventually, a reboot is required. That's why we wanted a solution that allows us to reboot servers without the need for a manual decryption step.

Of course, one could try to automate our existing process, letting another computer with access to the passphrase connect to the rebooting server via SSH and decrypt it. However, this type of automation is tricky, because it turns the whole thing into a distributed system, with all the problems that entails. We wanted a solution in NixOS that does not depend on another computer, but retains the security of our existing solution. And that's what we have achieved using **kexec**.

The kexec mechanism lets the kernel (k) execute another kernel (exec). Which means: The running kernel replaces itself with a new one without the actual server (i.e. the hardware) being shut down and restarted. Only the operating system restarts. The clever part: the old kernel can make information available in RAM that the new kernel can use. A passphrase for decrypting the hard drive, for example. As a nice side effect, we skip the firmware and bootloader parts of the reboot, saving several minutes of rebooting time, particularly on servers.

Securely passing passphrases

The LUKS passphrase could also be passed by simply writing it into a file before the reboot. But there's a catch: During a full reboot, the system shuts down completely. For the newly starting kernel to be able to read this file, it would have to be stored unencrypted, which would undermine the security of our hard drive encryption. With kexec, the data can be passed via the server's volatile RAM. This is a bit tricky though. In this article, we'll show you how we implemented it.

Security and convenience – both is possible

You can't have your cake and eat it, too? We simply didn't want to accept this supposed wisdom for the matter of rebooting our servers. We want all of it: the security of full-disk encryption, of course, but also convenience, speed and reliable reboots. And if there's one thing we don't do, it's giving up quickly just because something doesn't work straight away.

So we set out to find a solution and came across two interesting sources with different approaches:

- A comment in a lobste.rs thread.
- The blog article Encrypted NixOS home server with passwordless reboot.

Both approaches have their pros and cons; neither was quite enough for us. That's why we took the best bits from both and built our own solution.

Approach No. 1: One-time passphrases

We generate a temporarily valid passphrase. But we don't stop there. Because you can manage multiple keyslots in LUKS. So we also create a keyslot, assign this temporary

Approach No. 2: Do not pass the passphrase on the command line

Even though the passphrase is invalidated immediately on the next boot, we want to take extra care to ensure that nobody can read it from the kernel command line, not even in the brief moment it takes us to invalidate the one-time passphrase. We therefore use the technique described in the blog post mentioned above: the one-time passphrase is embedded in a special initramdisk image, which is created specifically just before the kexec reboot.

Our solution for passphrase-less reboots

And this is our specific solution: We extend `systemd.services."prepare-kexec"` with the following steps:

- We create a temporary LUKS passphrase in an additional key slot.
- We create a new initrd image and add a file containing the key (in RAM).
- We configure the use of this key file via the kernel command line and enable a fallback to manual passphrase entry.

Immediately after mounting the root filesystem, we remove the key slot via a custom systemd unit defined in `boot.initrd.systemd.services`.

Our uninterrupted 2-minute reboot today

Today, our reboot no longer requires any manual intervention. And it now takes just over 2 minutes. A simple `systemctl start kexec.target` is all that's needed. Whether triggered manually or automatically, the server comes back up fully functional and ready for use.

Passphrase-less reboots on NixOS to go

Would you like our complete implementation as ready-to-use code? You can find our code in a sample configuration below this article. Enjoy!

Sample NixOS module:

```
  luksDevice = config.boot.initrd.luks.devices."yourdevice".device;
in
{
  # Concept:
  # To reboot a server using kexec, we need to alter the nixos
  # prepare-kexec script, because we use full disk encryption.
  # We add a temporary, random key to LUKS keyslot 31. This key is
  # also added to the init ram disk image.
  # We have to set the keyfile and fallbackToPassword option in the
  # luks.device."yourdevice".If the keyfile exists it will be used
to
  # decrypt the disk, if not it will ask for the passphrase.
  # We immediately delete the LUKS slot after mounting.

  systemd.services."prepare-kexec" = {
    path = with pkgs; [ cpio cryptsetup gzip ];
    script = lib.mkForce ''
      set -euo pipefail
      umask 0077

      # Don't load the current system profile if we already have a
      # kernel loaded.
      if [[ 1 = "$(</sys/kernel/kexec_loaded)" ]] ; then
          echo "kexec kernel has already been loaded, prepare-kexec
skipped"
          exit 0
      fi

      p=$(readlink -f /nix/var/nix/profiles/system)
      if ! [[ -d $p ]]; then
        echo "Could not find system profile for prepare-kexec"
        exit 1
      fi

      if ! [[ -f "/LUKS-Passphrase-file.txt" ]]; then
        echo "Could not find luks-passphrase file"
        exit 1
      fi

      # add 256 random bytes temp key to the LUKS keyslot 31
      TEMP_DIR="$(mktemp -d --tmpdir=/dev/shm)"
      mkdir "$TEMP_DIR/etc"
```

```
file.txt

      # create a new cpio archive and append it to the original
initrd
      cd "$TEMP_DIR"
      cp "$p/initrd" "$TEMP_DIR/initrd.img"
      find etc | cpio -H newc -o | gzip >> "$TEMP_DIR/initrd.img"

      # load the kernel with the new initrd
      kexec --load "$p/kernel" --initrd="$TEMP_DIR/initrd.img" --
append="$(cat "$p/kernel-params") init=$p/init"
    '';
  };

  boot = {
    initrd = {
      luks.devices."yourdevice" = {
        fallbackToPassword = true;
        keyFile = "/etc/tmp-passphrase";
      };
      systemd.services.clear-luks-keyslot = {
        description = "Clear the LUKS key slot after successful
kexec";
        wantedBy = [ "initrd.target" ];
        after = [ "systemd-cryptsetup@root.service" ];
        serviceConfig.Type = "oneshot";
        path = with pkgs; [ cryptsetup ];
        script = "cryptsetup luksKillSlot --batch-mode
${luksDevice} 31 || true";
    };
  };
}
```

Germany

+49 221 2826780
mail@bevuta.com