Security

From NixOS Wiki
Jump to: navigation, search

Overview

This page is a guide to securing NixOS. Topics like hardening, process isolation, virtualization, firewalls, SELinux, containers, sandboxes, encryption, VPNs, etc. are in scope.

Core Nix features

These are security elements that are core features of using Nix(OS).

Obscurity of Nix store

In a vanilla NixOS system, the typical Linux filesystem hierarchy (https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard) is, in large part, replaced with the Nix store's user environments (/wiki/User_Environment). This means that some malware which might rely on finding system tools in particular places might fail. This is a form of security through obscurity (https://en.wikipedia.org/wiki/Security_through_obscurity) and is only a minor layer of protection.

Effort to isolate runtime search paths

In general, there is an effort to avoid rpath (https://en.wikipedia.org/wiki/Rpath) collisions across users [1] (https://github.com/NixOS/nix/commit/eba840c8a13b465ace90172ff76a0db2899ab11b).

Multi-user installation

NixOS is automatically installed in Multi-User mode. For standalone-Nix, the manual covers multi-user installs (https://nixos.org/manual/nix/stable/#ssec-multi-user). This allows multiple users to have isolated store environments and to avoid them having access to root in order to install their personal applications (achieved by having build users which nix operations are delegated to).

Data integrity and authenticity

The core installation resources for Nix(OS) have SHA256 (https://en.wikipedia.org/wiki/SHA-2) checksums which are GPG (https://en.wikipedia.org/wiki/GNU_Privacy_Guard) signed by the Nix team (https://nixos.org/download.html#nix-verify-installation) for authenticity. Within the installation data are all the SHA256 checksums for packages that were available within Nixpkgs at build time.

All packages which are pulled into your Nix system via Nixpkgs derivation builds are checked against SHA256 checksums which are already available on your local system (and should be traceable to the signed core Nix install materials).

Supported by Nix

These are features which are easily supported using Nix(OS).

Encryption

These are features which can protect data on a system.

Filesystem encryption

NixOS has LUKS (https://en.wikipedia.org/wiki/Linux_Unified_Key_Setup) partition-level disk encryption support.

- NixOS Manual - LUKS-Encrypted File Systems (https://nixos.org/manual/nixos/unstable/index.html#sec-luks-file-systems)
- Authenticated Boot and Disk Encryption on Linux (https://0pointer.net/blog/authenticated-boot-and-disk-encryption-on-linux.html)

Isolation

These are features which can limit a process or package's access to the host system.

Flatpaks

Flatpak (https://en.wikipedia.org/wiki/Flatpak)'ed applications are sandboxed (https://docs.flatpak.org/en/latest/sandbox-permissions.html) and require explicit privilege declaration for most access outside their own path. NixOS includes support for Flatpak (https://nixos.org/manual/nixos/unstable/index.html#module-services-flatpak). Note that, since Flatpak application dependencies are bundled/vendored (https://stackoverflow.com/questions/26217488/what-is-vendoring), this introduces other security risks (https://blogs.gentoo.org/mgorny/2021/02/19/the-modern-packagers-security-nightmare/) for the application . Also, most application flatpaks do no not make meaningful use of the sandbox (https://flatkill.org/).

Linux Containers

NixOS includes support for Linux Containers (LXC) (https://en.wikipedia.org/wiki/LXC). Containers, by default, do not provide much security. They are, oversimplifying a lot, a chroot (https://en.wikipedia.org/wiki/Chroot) environment with some resource constraints (cgroups (https://en.wikipedia.org/wiki/Cgroups)). The root user in a container would also be root on the whole system though. To avoid this, you must use *unprivileged containers*. There are some complications to this. The end of this post (https://blog.beardhatcode.be/2020/12/Declarative-Nixos-Containers.html) covers them well in brief.

References

- NixOS Manual - Administration: Containers Chapter (https://nixos.org/manual/nixos/unstable/index.html#ch-containers).
  - The manual — notably — currently has no mention of using unprivileged containers.
- LXC 1.0 Release, Security Features Coverage (user namespaces, unprivileged containers) (https://stgraber.org/2014/01/01/lxc-1-0-security-features/)
- Youtube - Red Hat: How containers use PID namespaces to provide process isolation (https://www.youtube.com/watch?v=J17rXQ5XkDE)
- What Are Namespaces and cgroups, and How Do They Work? (https://www.nginx.com/blog/what-are-namespaces-cgroups-how-do-they-work/)

- A Tutorial for Isolating Your System with Linux Namespaces (code-based fundamental examples) (https://www.toptal.com/linux/separation-anxiety-isolating-your-system-with-linux-namespaces)

Docker Containers

Docker (https://en.wikipedia.org/wiki/Docker_(software)) is a system for building and running platform-independent virtual containers. On Linux, it is implemented similarly to LXC. Nix integrates tools to create Docker images (the templates for making Docker containers), documented in the Nixpkgs manual (https://nixos.org/manual/nixpkgs/unstable/#sec-pkgs-dockerTools). Docker containers work with namespacing controls (https://docs.docker.com/engine/security/) similar to unprivileged LXC containers by default.

Virtual machines

Virtual machines (https://en.wikipedia.org/wiki/Virtual_machine) are generally one of the most robust tools available for process isolation. They come with performance penalties (https://www.brendangregg.com/blog/2017-11-29/aws-ec2-virtualization-2017.html) and resource overheads.

NixOS includes support for hosting virtual machines. The Nix store of the host machine is shared read-only with guest machines, making them lighter-weight in terms of storage use than typical VMs. Guest VMs are easily built from Nix configurations.

- qemu-vm.nix - Implementation of QEMU builds of NixOS machine configurations (https://github.com/NixOS/nixpkgs/blob/master/nixos/modules/virtualisation/qemu-vm.nix)
- Firecracker microVM (https://github.com/firecracker-microvm/firecracker) - packages available (https://search.nixos.org/packages?channel=unstable&show=firecracker&from=0&size=30&sort=relevance&type=packages&query=firecracker)
- microvm.nix (https://github.com/astro/microvm.nix) builds and runs NixOS on various hypervisors

Test machines

NixOS integrates support for building test VMs (/wiki/NixOS:nixos-rebuild_build-vm) to test configuration changes to your system.

NixOS also uses VMs to continuously validate functionality of the system. NixOS Manual - Writing Tests Chapter (https://nixos.org/manual/nixos/unstable/index.html#sec-writing-nixos-tests) covers this.

References

- Updating NixOS local VMs (http://blog.patapon.info/nixos-local-vm/) - Post demonstrating how to build and run a VM from a NixOS configuration and then update the configuration of the running VM on the fly.
- 2020 IBM Presentation on Address Space Isolation in the Linux Kernel (https://archive.fosdem.org/2020/schedule/event/kernel_address_space_isolation/attachments/slides/3889/export/events/attachments/kernel_address_space_isolation/slides/3889/Address_Space_Isolation_in_the_Linux_Kernel.pdf) - Containers within VMs are a norm for security in the cloud. Addressing ongoing work to improve isolation of containers and VMs.
- An EPYC escape: Case-study of a KVM breakout (https://googleprojectzero.blogspot.com/2021/06/an-epyc-escape-case-study-of-kvm.html) - Detailing first known non-userspace vulnerability enabling guest-to-host breakout.

Systemd Hardening

It is possible to increase the isolation of Systemd services by using hardening options. For example, adding `PrivateNetwork=yes` option in a Systemd unit removes access to the host network.

For more information, please see Systemd_Hardening (/wiki/Systemd_Hardening).

Networking

These are features which improve system security in relation to the world outside of the system.

Firewall

NixOS includes a simple stateful firewall. It blocks incoming connections and other unexpected packets and is enabled by default.

- NixOS Manual - Networking Chapter: Firewall (https://nixos.org/manual/nixos/unstable/index.html#sec-firewall)
- NixOS Manual - Options: networking.firewall.enable (https://nixos.org/manual/nixos/unstable/options.html#opt-networking.firewall.enable)

Awaiting NixOS support

This section covers important security technologies which still need full NixOS support.

Secure Boot

Development of UEFI Secure Boot (https://en.wikipedia.org/wiki/Unified_Extensible_Firmware_Interface#Secure_Boot) support is in flight (https://github.com/NixOS/nixpkgs/issues/42127). An experimental secure boot implementation is available in Lanzaboote (https://github.com/nix-community/lanzaboote)

SELinux

It is possible to use Security-Enhanced Linux (SELinux) (https://en.wikipedia.org/wiki/Security-Enhanced_Linux) in NixOS, but proper integration does not exist. This does not appear to have gotten much attention since 2019 (https://github.com/NixOS/rfcs/pull/41).

Nix official references

- Nix Manual - Security Chapter (https://nixos.org/manual/nix/stable/#ch-nix-security)

See also

NixOS

- Blog - Paranoid NixOS Setup (https://christine.website/blog/paranoid-nixos-2021-07-18)
- vulnix (https://github.com/flyingcircusio/vulnix) - Vulnerability (CVE) scanner for Nix/NixOS

Projects leveraging Nix for security

- Spectrum OS (https://spectrum-os.org/) - Nix based, Spectrum is a project that aims to create a computer operating system, based on the principle of security by compartmentalization, that has a lower barrier to entry and is easier to use and maintain than other such systems.

General Linux hardening

- The Practical Linux Hardening Guide (https://github.com/trimstray/the-practical-linux-hardening-guide)

- How To Secure A Linux Server (https://github.com/imthenachoman/How-To-Secure-A-Linux-Server) - An evolving how-to guide for securing a Linux server that, hopefully, also teaches you a little about security and why it matters.
- lynis (https://github.com/CISOfy/lynis) - Security auditing tool for Linux, macOS, and UNIX-based systems. Assists with compliance testing (HIPAA/ISO27001/PCI DSS) and system hardening. Agentless, and installation optional.
- DevSec Linux Baseline (https://github.com/dev-sec/linux-baseline) - InSpec profile to test Linux server security config

General systems hardening

- awesome-security-hardening (https://github.com/decalage2/awesome-security-hardening) - Collection of security hardening guides, tools and other resources.

Retrieved from "https://nixos.wiki/index.php?title=Security&oldid=12720 (https://nixos.wiki/index.php?title=Security&oldid=12720)"

Categories (/wiki/Special:Categories): Guide (/wiki/Category:Guide) | NixOS (/wiki/Category:NixOS) | Nix (/wiki/Category:Nix)