White Paper

Nutanix AHV

Security at the Virtualization Layer

NUTANIX

Table of Contents

Introduction

Nutanix Acropolis Hypervisor (AHV) is a modern and secure virtualization platform that provides virtualization capabilities in powering VMs and Kubernetes® containers for applications and cloud-native workloads on-premises and in public clouds. AHV is a hypervisor designed to run multiple workloads with the protection of sensitive data. Some techniques used included restricting access to data, VM isolation, or segmenting the underlying machines running the workloads. One of the benefits of virtualization is security; applications running in separate virtual machines are isolated from each other and the infrastructure running those virtual machines. AHV leverages hardware virtualization technologies, allowing multiple operating systems or containers to run concurrently on a single physical machine. AHV leverages Intel's Virtualization Technology (VT-x) or AMD's AMD-V, the underlying technology supporting the simultaneous execution of multiple operating systems and containers in a secure, isolated virtualized environment.

What is AHV?

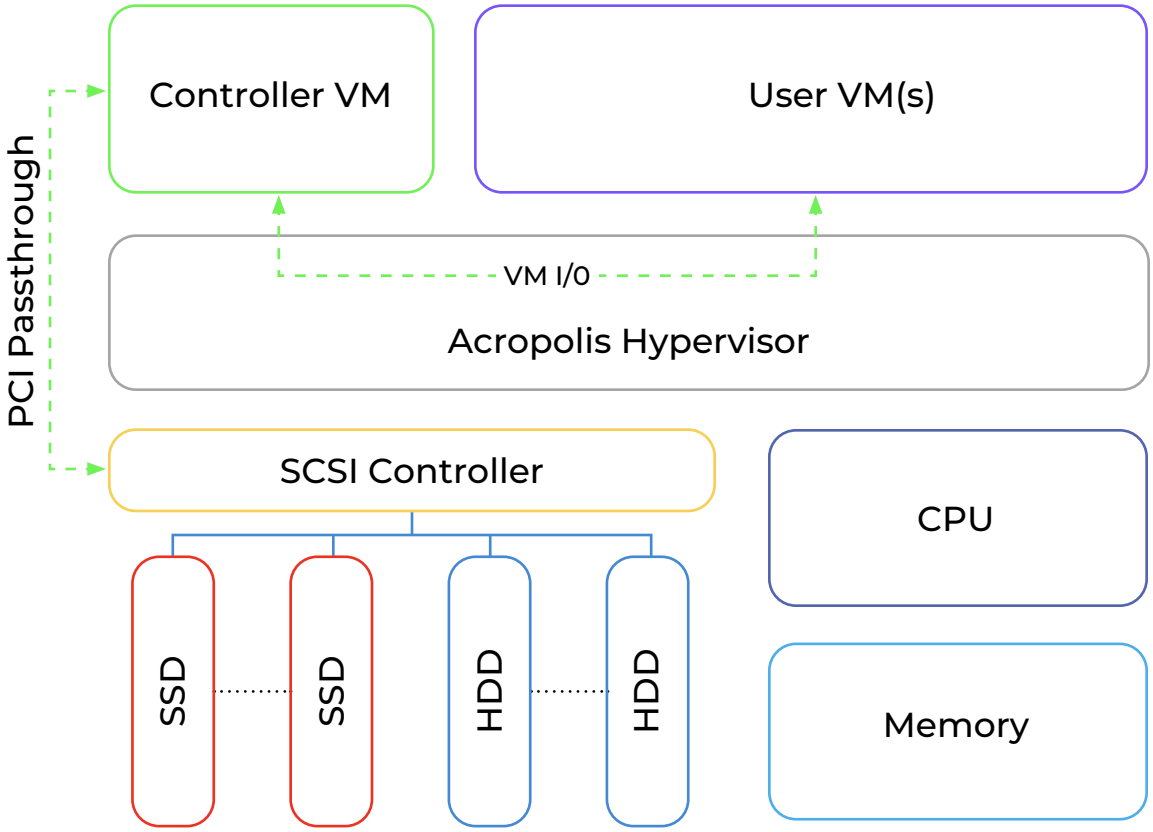
AHV is an enterprise-class virtualization solution included with the Nutanix Acropolis Operating System (AOS), which is included in the Nutanix Cloud Infrastructure (NCI) license. AHV, as a Type 1 (native) hypervisor, comes preinstalled on Nutanix appliances, is configurable in minutes, and provides all the necessary foundation for your virtual infrastructure. AHV was designed and optimized for the modern Cloud era, built for Hyperconverged Infrastructure (HCI) or Compute Only deployments, based on the principles of web-scale engineering, delivering Operational Intelligence, Security, and Automation with 1-click simplicity.

To provide some context, Nutanix introduced KVM support in 2013 with AOS 3.5, having already supported ESX. This was a precursor to the launch of our native hypervisor, i.e., AHV, which was formally released with AOS 4.1 in 2015. Under the covers, AHV is an adaptation of the proven open-source Linux KVM and QEMU. It subsequently was extended with a number of enhancements focused on optimizing the I/O path, memory management, scaling, and security.

AHV is more than a branded version of Linux KVM virtualization; Nutanix has invested in creating a virtualization solution that is as easy to manage as our industry-leading HCI offering, providing all the performance and enterprise-grade features needed to run any workload.

For more information on AHV, see The Nutanix Cloud Bible for AHV.

AHV Architecture that is built to be Secure



AHV Security begins with architecture, where AHV engineering prioritizes design decisions that are best for the hypervisor's overall functionality and builds a secure foundation. The goal of a hypervisor is isolation: between virtual machines, the external world, and the hypervisor itself. Each of those pairs contributes to overall security:

- **Virtual machines / virtual machines:** the hypervisor presents each virtual machine with an illusion of sole ownership of a set of resources (compute, storage, network). This is the definition of "virtualization": presenting resources as exclusive while in reality they are shared.
- **Virtual machines / hypervisor:** the hypervisor allows management of the virtual machines only through a control plane interface, which is not available to the virtual machines. This allows Nutanix administrators to control their workloads with predictability, independent of the content of the workload.
- **Hypervisor / external world:** the hypervisor protects its own resources from unauthorized access, ultimately protecting the workloads running on the hypervisor.
- **Virtual machines / external world:** the hypervisor provides hooks for value-added security that would not be possible or cost-effective without running in a virtualized environment. This is especially visible with technologies like network microsegmentation and application firewalls.

Virtual Machine Integrity

Compute

AHV utilizes the **Kernel-based Virtual Machine** (KVM) and **Quick Emulator** (QEMU) to run virtual machines. This requires CPU support from **Intel VT-x** or **AMD-V** (also called SVM), which provides an isolated execution environment for the virtual machine. The hypervisor arranges to periodically interrupt the virtual machine's execution to retain scheduler control, allowing the hypervisor to run the control plane and other virtual machines simultaneously. This scheduling is transparent to the virtual machine. When the virtual machine needs to access a resource unavailable within this isolated environment, the hypervisor regains control and substitutes its access to that resource, typically storage or networking. The usage of hardware virtualization instructions prevents a virtual machine from accessing the resources of the hypervisor or other virtual machines, and the hypervisor's scheduling choices help ensure a virtual machine cannot excessively use resources to inhibit the availability of the hypervisor or other virtual machines.

AHV utilizes **Intel Extended Page Tables** (EPT) or **AMD Nested Page Tables** (NPT, also called RVI) to virtualize the memory assigned to a virtual machine efficiently. When combined with an IOMMU such as **Intel VT-d** or **AMD-Vi**, AHV can pass through any PCI device directly to a virtual machine while restricting the access of that PCI device to only the virtual machine's resources. This is used for NVIDIA GRID GPU passthrough.

Storage

AHV is part of the **Nutanix Hyper-Converged Infrastructure**, where the Controller Virtual Machine (CVM) directly controls any attached storage, converts it into a storage cluster, and provides virtualized storage access using AOS vDisk. AHV is designed to help virtual machines exclusively use this CVM-provided storage for all storage needs. This helps ensure that virtual machines are solely paired with their configured vDisk and do not have access to the storage resources of other virtual machines or the hypervisor, and that the Nutanix administrator retains complete control of storage allocation and utilization.

Network

AHV utilizes **Open vSwitch** (OVS) within the hypervisor so that the control plane can configure network routes and topologies outside the virtual machine's control. OVS by default functions as an L2 switch (IP), which enables the hypervisor to restrict virtual machines only to the OVS switch configured by the administrator. This helps restrict network access to only

the endpoints the administrator has configured for that OVS, including attached virtual machines and external egress points.

The Open vSwitch primitive allows more sophisticated processing, which the **Nutanix Flow** product utilizes to provide further security primitives. Flow utilizes OVS to provide L4 firewall capabilities, including policy-based microsegmentation at a per-connection granularity based on its knowledge of the workloads being run on the hypervisor and configuration of **Virtual Private Clouds** (VPCs). Flow can also direct traffic through packet inspection, stateful firewalls, NATs, and VPNs to provide even more powerful security capabilities, all supported by the AHV hypervisor, which helps ensure Flow's network processing is present on its virtual machines.

Guest Operating System

Some features of AHV are designed to allow the guest operating system to better secure itself by utilizing hypervisor support.

- **Secure Boot** in the virtual machine's UEFI firmware will verify code signatures of the operating system as it boots. This is designed to allow the virtual machine to boot only an intended operating system and provides protection against operating system tampering.
- **Virtual TPM (vTPM)** is an emulated device that implements a TPM device for each virtual machine. AHV will have a private vTPM instance for each VM guest. vTPM allows each VM guest to have its own key storage, isolating it from the other guests running on the same physical server. TPMs provide secure key storage, random number generation, authentication, and attestation. These capabilities are implemented outside the control of the guest operating system and under the control of the hypervisor. This allows the guest operating system to trust that the TPM capabilities are secure and allows the guest operating system to prove security claims to third parties. Because they are under hypervisor control, the hypervisor can migrate them along with the virtual machine within NCI. Example security behaviors the vTPM enables:
    - Secret sealing: a guest can seal a secret to help ensure it can only be unsealed when run on Nutanix NCI connected to the same secret stores. Guest operating systems often use this to implement disk encryption or secret stores, for example, Microsoft BitLocker or Windows Hello, to help protect data or multi-factor authentication from being stolen or cloned.
    - Attestation quoting: a guest can use the vTPM to sign a record of the software that was booted, and can thus prove to a 3rd party that it is running the correct, untampered software. In effect, AHV acts as a notary to prove the guest identity.

- **Hypervisor-protected Code Integrity (HVCI)** is a feature of Microsoft Windows that isolates security functionality from the Windows kernel by asking the hypervisor to run parts of the Windows kernel in separate mini-VMs. This protects Windows against rootkits and other forms of malware that would usually hide within the kernel by bypassing Windows security checks. HVCI requires a simplified form of nested virtualization (running Hyper-V as a VM inside an AHV VM), which AHV provides, and helps protect against certain categories of malware, especially targeting the ability of malware to hide itself. Example security behaviors of HVCI:
  - Device driver code signature checks are run by HVCI, ensuring the signature check cannot be bypassed.
  - Memory protections are changeable only by HVCI, helping protect against some forms of malware from attacking the kernel.
- **Generation ID** allows a virtual machine to detect that it was cloned by a hypervisor. This is important for generating unique IDs, especially IDs that must be unique for security purposes. For example, a Windows system is identified by a Security Identifier (SID), and two virtual machines using the same SID would allow two systems to impersonate each other. The Generation ID allows the guest operating system to detect this duplication and rotate identifiers.

Hypervisor Integrity

Install and Boot

Nutanix builds AHV images with code signatures, which prove the software was created by Nutanix and has not been tampered with since being built. NCI verifies the AHV code signature when loading an AHV image to install, which is designed to help prevent deployment of tampered software. When Secure Boot is enabled on the hardware side, UEFI logic will verify that the AHV binaries are trusted and have not been tampered with.

AHV is designed to be secure and hardened by default, delivering a wide range of capabilities. During the installation of AHV, some additional hardening is required where the installer requires information. See the Nutanix Security Guide in our documentation portal for more information.

AHV adopts a “stateless” design philosophy where all virtual machine state is stored at the cluster level by the CVM. This avoids storing state that could be tampered with while AHV is not running, and the complexity of confirming state is still secure, mitigating persistent attacks that hide in tampered state. This stateless design also makes installing patches and updates more robust by avoiding the risk of incorrectly upgrading state. It makes restoring trust in an AHV system easier: the difference between a reboot and a reimage is minimal.

The configuration of AHV focuses on eliminating external access. The hypervisor is managed through the CVM (running on AHV). Direct access to AHV is limited to emergency troubleshooting.

Runtime

AHV uses compiler hardening and execution mitigation techniques to reduce the risk of implementation errors.

- **Usage of memory-safe languages:** Much of the AHV functionality, especially control plane functionality, is implemented in Rust or Go or Python, languages that eliminate memory buffer attacks common to C and C++.
    - AHV does not make use of or contain Java.
- **Address Space Layout Randomization** (ASLR) is a technique to randomize addresses and typically increases the difficulty of an attack by requiring a second vulnerability to leak addresses. AHV enables ASLR for all binaries.
- **Non-Executable** (NX) protection is a technique that avoids leaving memory both writable and executable. AHV binaries are compiled to allow NX protection.
- **Spectre mitigations** are a set of techniques to prevent information leakage through speculative execution. AHV employs these techniques by default when the performance impact is nominal, and offers configuration options to meet a customer's risk profile when the performance impact is substantial.

NCI's hyperconverged infrastructure deploys a "backplane" for communication within a cluster as a configuration in AHV's OVS networking. CVMs communicate with other CVMs and AHVs in the same cluster over this backplane, which is routable only to CVMs and AHVs in the cluster.

For more information on AHV architecture, please refer to AHV Architecture in the Nutanix Bible.

Governance

AHV has adopted recommendations from the Defense Information Systems Agency (DISA) in establishing cybersecurity configuration standards to minimize vulnerabilities and improve security. AHV has implemented hundreds of comprehensive sets of security controls outlined in the **Security Technical Implementation Guide (STiGS)**, which are configuration recommendations from DISA. The US Department of Defense (DoD) has confirmed our implementation of these required STIGS, with Nutanix receiving a listing on the DoD Approved Product list certification.

Windows Guest on AHV

Today, hardware manufacturers offer several security protections for the running Operating Systems (OS), such as Unified Extensible Firmware Interface(UEFI), Secure Boot, and Trusted Platform Module (TPM), all discussed above. These hardware technologies are designed to protect the underlying system being booted and are not necessarily designed to protect VM guests running in a virtualized environment.

Microsoft has invested in several areas leveraging these technologies such as UEFI, Secure Boot, and TPM technologies.

- **Microsoft Windows 11** – requires UEFI, Secure Boot, and TPM technologies.
- **Microsoft Credential Guard** requires UEFI and Secure Boot and optionally will leverage the TPM technologies in protecting the key used by Microsoft Virtual Base Security.
- **Microsoft Bitlocker** optionally will leverage the TPM technologies to protect its keys and encrypt all the user and system files on the Windows drive.
- **The Linux** community can optionally leverage UEFI and Secure Boot.

AHV offers these protections, as discussed earlier. They are built into the hypervisor and are heavily influenced by Microsoft's latest release of Windows, which requires UEFI, Secure Boot, and TPM.

Microsoft Windows 11 documentation outlines these requirements and provides detailed information on leveraging these capabilities. Be aware that Microsoft Windows documentation does not distinguish between running on a bare machine and running Windows in a hypervisor environment. The Windows guest OS is unaware of the underlying hypervisor's virtual implementation of UEFI, Secure Boot, and TPM. These features are implemented in the hypervisor, which is not required on the hardware.

In a virtual environment, a hypervisor such as AHV allows multiple guests to run on a single hardware device sharing the hardware resources for the running guests. Hardware security technologies, such as UEFI, Secure Boot, and TPM, offer security protection between the hardware and the running hypervisor but do not directly benefit the VM running within the hypervisor. Each hypervisor vendor has implemented a hypervisor software version extending the security benefits to the VM Guest.

Nutanix Secure Development Life Cycle

Security is embedded in the complete life cycle of Nutanix engineering, from requirements gathering to design, development, testing, and delivery. Secure coding is a principal to the coding process. Code Analysis techniques, Vulnerability scanning, Threat Modelling, image and artifact scanning, AV scanning, and rigorous Penetration testing have been used by Nutanix as part of its efforts to help secure the hypervisor.

Nutanix is dedicated to providing secure products to our customers. For up-to-date information on Security Advisories and CVE fixes, check https://www.nutanix.com/trust/security-advisories.

Conclusion

Nutanix AHV is designed to be secure by default, offering comprehensive security capabilities. AHV is just one infrastructure component that operates as part of the Nutanix Cloud Platform; safeguards exist throughout PRISM, AOS, and AHV in combination, which have not been fully explored in this paper. Nutanix products and documented best practices are available to protect the data plane, such as, Data-at-Rest Encryption, Role-Based Access Control, Secure Snapshot, and more. Nutanix recommends reviewing our security guide available in the Nutanix documentation portal. Additional information about Nutanix security can be found at Nutanix Trust.

NUTANIX

info@nutanix.com | www.nutanix.com | @nutanix

PSM-Nutanix-AHV-A-Secure Hypervisor-White-Paper-FY26Q4