The Security Design of the AWS Nitro System

AWS Whitepaper

The Security Design of the AWS Nitro System: AWS Whitepaper

Table of Contents

The Security Design of the AWS Nitro System

Publication date: **November 18, 2022** (*Document revisions* (p. 27))

Abstract

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. The AWS Nitro System is the underlying platform for all modern EC2 instances. This whitepaper provides a detailed description of the security design of the Nitro System to assist you in evaluating EC2 for your sensitive workloads.

Introduction

Every day, customers around the world entrust Amazon Web Services (AWS) with their most sensitive applications. At AWS, keeping our customers' workloads secure and confidential, while helping them meet their security, privacy, and data protection requirements, is our highest priority. We've invested in rigorous operational practices and security technologies that meet and exceed even our most demanding customers' data security needs.

The development of the AWS Nitro System has been a multi-year journey to reinvent the fundamental virtualization infrastructure of Amazon EC2. Since launching the Amazon EC2 beta in 2006, we continued to refine, optimize, and innovate in all facets of the service to meet the needs of our customers. With the AWS Nitro System, we undertook an effort to dramatically reimagine the architecture of virtualization to deliver the security, isolation, performance, cost, and pace of innovation that our customers require.

Security has been a fundamental principle of that process from day one, and we continued to invest in the implementation of the design as part of our continuous improvement methodology to keep raising the bar of security and data protection for our customers. The AWS Nitro System is a combination of purpose-built server designs, data processors, system management components, and specialized firmware which provide the underlying platform for all Amazon EC2 instances launched since the beginning of 2018. Together, the limited and discretely designed components of the AWS Nitro System deliver faster innovation, enhanced security, and improved performance for EC2 customers.

Three key components of the Nitro System achieve these goals:

- **Purpose-built Nitro Cards** — Hardware devices designed by AWS that provide overall system control and input/output (I/O) virtualization independent of the main system board with its CPUs and memory.
- **The Nitro Security Chip** — Enables a secure boot process for the overall system based on a hardware root of trust, the ability to offer bare metal instances, as well as defense in depth that offers protection to the server from unauthorized modification of system firmware.
- **The Nitro Hypervisor** — A deliberately minimized and firmware-like hypervisor designed to provide strong resource isolation, and performance that is nearly indistinguishable from a bare metal server.

Note
These components are complementary but do not need to be used together.

This paper provides a high-level introduction to virtualization and the fundamental architectural change introduced by the Nitro System. It discusses each of the three key components of the Nitro System, and provides a demonstration of how these components work together by walking through what happens when a new Amazon Elastic Block Store (Amazon EBS) volume is added to a running EC2 instance. The whitepaper discusses how the Nitro System, by design, eliminates the possibility of administrator access to an EC2 server, the overall passive communications design of the Nitro System, and the Nitro System change management process. Finally, the paper surveys important aspects of the EC2 system design that provide mitigations against potential side-channels issues that can arise in compute environments.

Traditional virtualization primer

Virtualization, at a high level, enables a single physical computer system to run multiple operating systems at once. A virtualization system ("host") implements translation, emulation, and restriction functions that allow it to provide one or more virtualized operating systems ("guest") with virtual representations of their own self-contained hardware ("virtual machines" or "VMs"). In other words, a virtualization host. One of the primary benefits of virtualization is the ability to make efficient use of a single powerful server by dividing its resources among multiple virtual machines each of which is allocated an amount of resources which is optimal for its assigned tasks.

Note

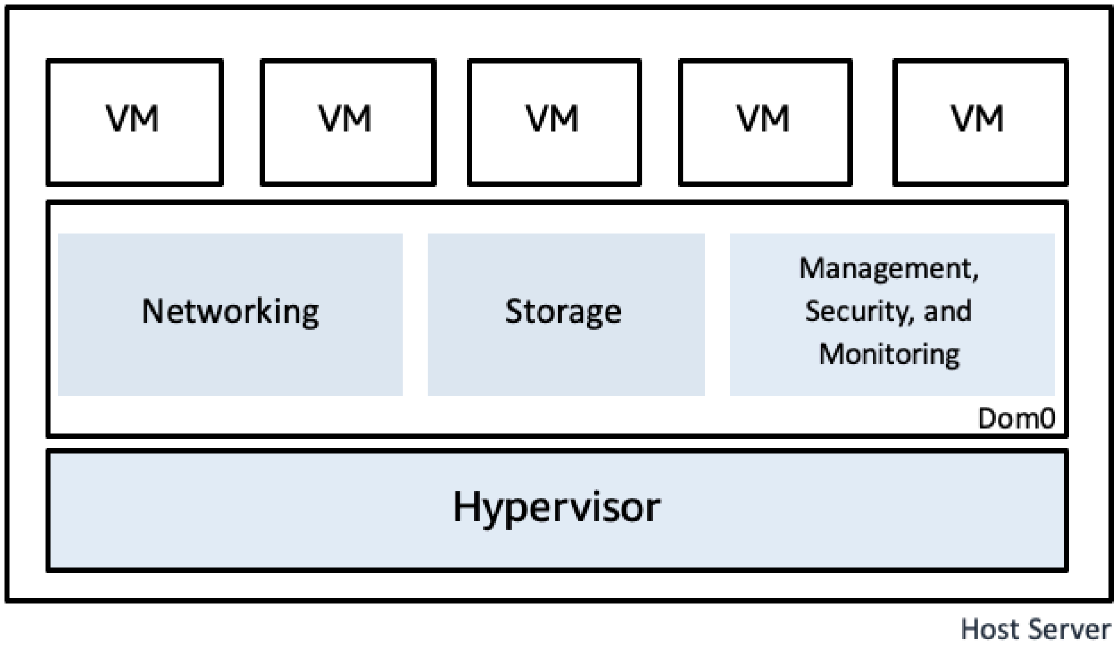
The discussion of virtualization in this section provides a generalized high-level introduction to the topic and does not dive into topics such as Paravirtualization, in which guest software must be modified to run in the virtualized environment. For a more detailed primer on virtualization technologies, refer to this presentation on virtualization technologies by Anthony Liguori, VP and Distinguished Engineer at AWS.

The core component responsible for managing the lifecycle and operation of guest virtual machines (VMs) in a virtualization system is called a virtual machine monitor (VMM), or *hypervisor*. For a statistical majority of operations it performs, a guest runs instructions natively on the system's physical CPU without any involvement by the VMM. For example, when a guest seeks to compute the sum or product of two values, it can communicate directly with the CPU hardware of the system to issue the requisite machine code instructions.

There are, however, some classes of sensitive or privileged instructions, such as reading or writing from CPU control registers, that a guest should not be allowed to run directly on the CPU hardware to maintain the stability and isolation of the system as a whole. When a guest tries to issue one of these instructions to the CPU, instead of running, the instruction is redirected to the VMM, which emulates a permitted result for the instruction, and then returns control back to the guest as if the instruction had been performed on the CPU natively.

A VMM itself is a relatively simple piece of software. However, a virtualization host requires more than the core functionality of a VMM to provide guests with access to devices such as network interfaces, storage drives, and input peripherals. To provide these features, hosts rely on additional software components called *device models*. Device models communicate with the system's shared physical I/O hardware and emulate the state and behavior of one or more unique virtual device interfaces exposed to guest VMs.

Hypervisors typically employ a general-purpose operating system to interface with a variety of system hardware, run device models, and run other management software for the virtualization system. This operating system is commonly implemented as a special privileged virtual machine which, for example, the Xen Project calls the system's dom0, and Hyper-V calls the system's root/parent partition. In early generation EC2 instances, this took the form of a special Amazon Linux VM running as what in Xen terminology is called *domain 0*, or *dom0*.

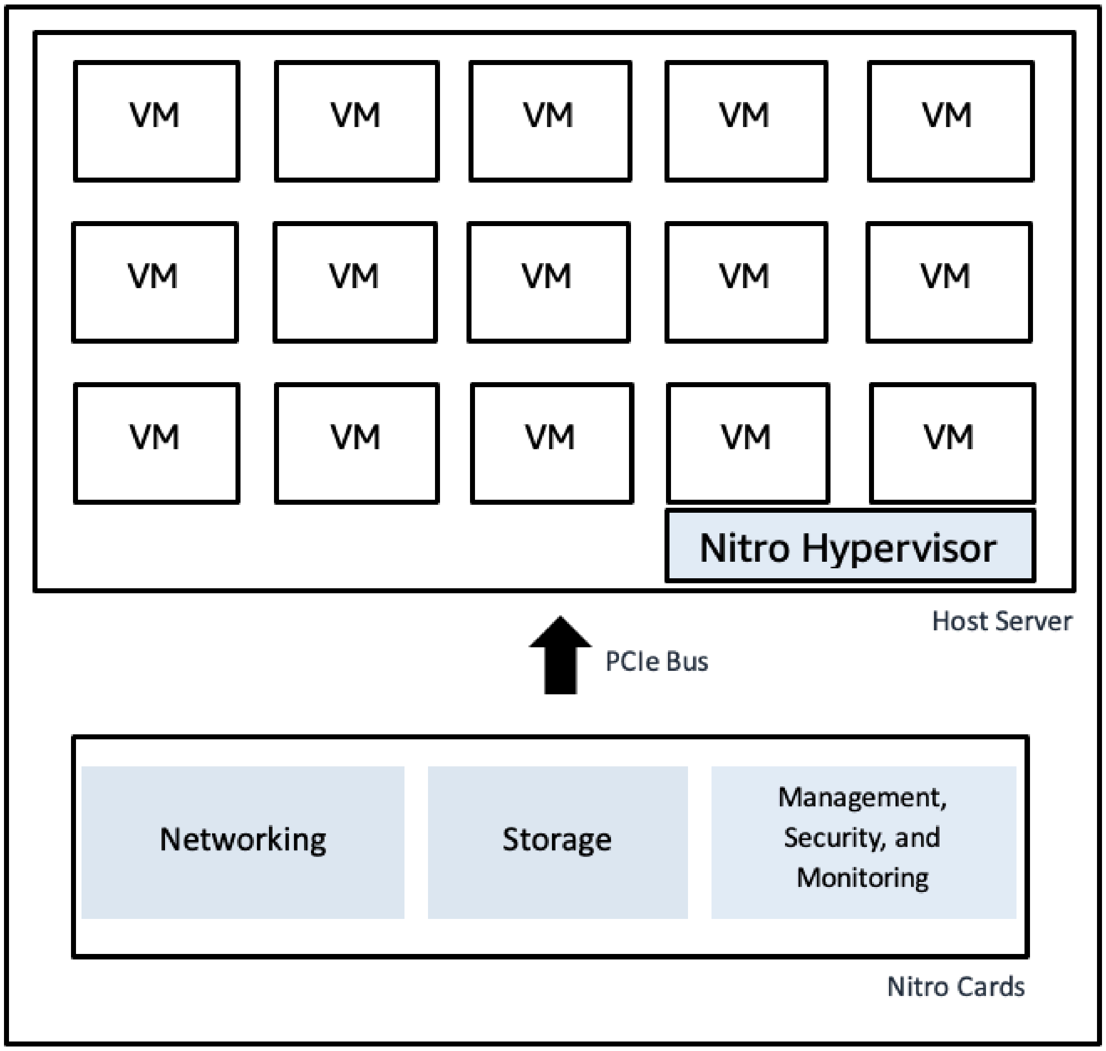


Classical virtualization architecture

The Nitro System journey

The Nitro System is the product of a multi-year journey of re-imagining virtualization technology for AWS Cloud infrastructure. Over the course of this journey, every component of virtualization technology was re-implemented and replaced. While customers saw improved cost, performance, and security from EC2 instances released earlier in this process, instances based on the resulting complete Nitro System, in which every component has been replaced, are meaningfully different from those prior instance types. The Nitro System provides enhanced security, confidentiality, and performance to customers of Amazon EC2, and provides a foundation that enables the delivery of new innovative technologies at a rapid pace.

The introduction of the Nitro System consisted of an incremental decomposition of the software components running in Dom0 on a general-purpose data center CPU into independent purpose-built co-processor units. What started as a tightly coupled monolithic virtualization system was, step by step, transformed into a purpose-built microservices architecture. Starting with the C5 instance type introduced in 2017, the Nitro System has entirely eliminated the need for Dom0 on an EC2 instance. Instead, a custom-developed, minimized hypervisor based on KVM provides a lightweight VMM, while offloading other functions such as those previously performed by the device-models in Dom0 into a set of discrete Nitro Cards.



Nitro System virtualization architecture

The components of the Nitro System

As noted earlier, the Nitro System consists of three primary components:

- Purpose-built Nitro Cards
- The Nitro Security Chip
- The Nitro Hypervisor

The Nitro Cards

A modern EC2 server is made up of a main system board and one or more Nitro Cards. The system main board contains the host CPUs (Intel, AMD, or Graviton processors) and memory. *Nitro Cards* are dedicated hardware components with powerful 64-bit processing capabilities and specialized Application Specific Integrated Circuits ("ASICs") that operate independently from the system main board that runs all customer compute environments, including code and data processing operations.

The Nitro Cards implement all the outward-facing control interfaces used by the EC2 service to provision and manage compute, memory, and storage. They also provide all I/O interfaces, such as those needed to provide software-defined networking, Amazon EBS storage, and instance storage. This means that all the components that interact with the outside world of the EC2 service beyond the main system board—whether logically inbound or outbound—run on self-contained computing devices which are physically separate from the system main board on which customer workloads run.

The Nitro System is designed to provide strong logical isolation between the host components and the Nitro Cards, and this physical isolation between the two provides a firm and reliable boundary which contributes to that design. While logically isolated and physically separate, Nitro Cards typically are contained within the same physical server enclosure as a host's system main board and share its power supply, along with its PCIe interface.

Note
In the case of the EC2 mac1.metal and mac2.metal instances. A Nitro Controller is colocated with a Mac Mini in a common metal enclosure, and the two are connected together with Thunderbolt. Refer to Amazon EC2 Mac Instances.

The main components of the Nitro Cards are AWS-designed System on a Chip (SoC) package that run purpose-built firmware. AWS has carefully driven the design and implementation process of the hardware and firmware of these cards. The hardware is designed from the ground up by Annapurna Labs, the team responsible for the AWS in-house silicon designs. The firmware for these cards is developed and maintained by dedicated AWS engineering teams.

Note
Annapurna Labs was acquired by Amazon in 2015 after a successful partnership in the initial phases of the development of key AWS Nitro System technologies. Annapurna is responsible not only for making AWS Nitro System hardware, but also for the AWS custom Arm-based Graviton processors, the AWS Tranium and AWS Inferentia hardware accelerator chips for ML training and inference, AWS Nitro SSD, and Aqua (advanced query accelerator) for Amazon Redshift.

The critical control firmware on the Nitro Cards can be live-updated, using cryptographically signed software packages. Nitro Cards can be updated independently of other components of the Nitro System, including of one another and of any updateable components on the system main board in order to deploy new features and security updates. Nitro Cards update with nearly imperceptible impact on customer workloads and without the relaxing of any of the security controls of the Nitro System.

The Nitro Cards are physically connected to the system main board and its processors via PCIe, but are otherwise logically isolated from the system main board that runs customer workloads. A Nitro System can contain one or more Nitro Cards; if there is more than one, they are connected through an internal network within a server enclosure. This network provides a private communication channel between Nitro Cards that is independent of the system mainboard, as well as a private connection to the Baseboard Management Controller (BMC), if one is present in the server design.

The Nitro Controller

The primary Nitro Card is called the Nitro Controller. The Nitro Controller provides the hardware root of trust for the overall system. It is responsible for managing all other components of the server system, including the firmware loaded onto the other components of the system. Firmware for the system as a whole is stored on an encrypted solid state drive (SSD) that is attached directly to the Nitro Controller. The encryption key for the SSD is designed to be protected by the combination of a Trusted Platform Module (TPM) and the secure boot features of the SoC. This section describes the secure boot design for the Nitro Controller and its role as the trusted interface between a server and the network.

Note
In the case of AWS Outpost deployments, a Nitro Security Key is also used along with a TPM and the secure boot features of the SoC to protect the encryption key for the SSD, which is connected directly to the Nitro Controller.

The Nitro Controller secure boot design

The secure boot process of the SoC in the Nitro Controller starts with its boot ROM and then extends a chain of trust by measuring and verifying early stages firmware stored in flash attached to the Nitro Controller. As the system initialization progresses, a trusted platform module (TPM) is used to record the initial boot code measurements, and then to extend measurements to additional system firmware. The cryptographic keys embedded in the tamper-resistant TPM are used to digitally sign the complete set of known good system measurements. This digitally signed file is then compared to all subsequent system measurements at each reboot to detect any unexpected changes.

If no changes are detected, additional decryption keys encrypted by keys locked in the TPM are used to decrypt additional data in the system to allow the boot process to continue. If changes are detected, the additional data is not decrypted and the system is immediately removed from service and will therefore not host customer workloads.

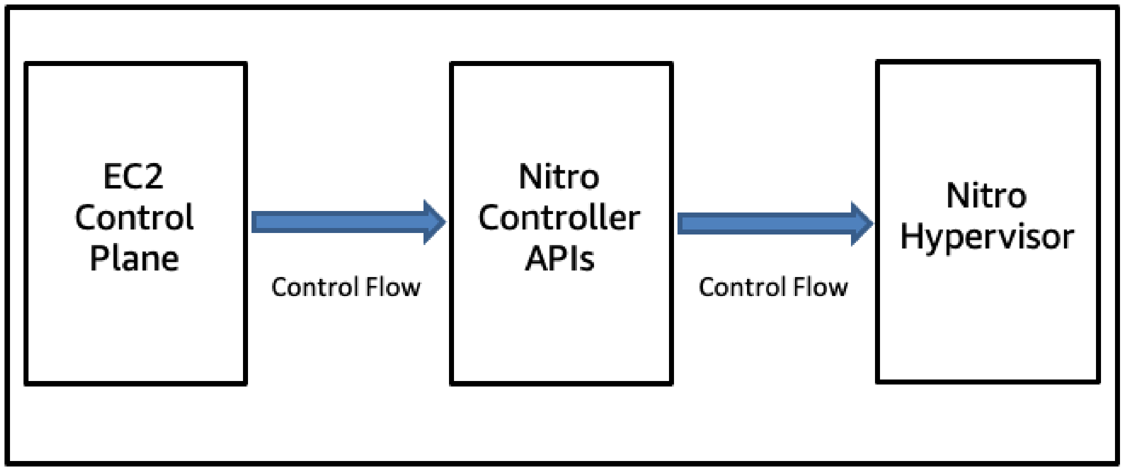
The preceding steps detail the process by which the Nitro Controller establishes the integrity and validity of system software on boot. For secure boot design to be truly secure, each stage of SoC boot code must not only be valid and unmodified, but also functionally correct as implemented. This is especially true of the static ROM code that is a part of the physical manufacturing of the device. To that end, AWS has applied formal methods to verify the memory safety properties of early-stage boot code in our designs.

The Nitro Controller as interface between EC2 servers and the network

The Nitro Controller is the exclusive gateway between the physical server and the control planes for EC2, Amazon EBS, and Amazon Virtual Private Cloud (Amazon VPC). While logically distinct, and composed of multiple sub-component microservices, these three control planes are hereafter generally referred to as the *EC2 control plane*.

Note
Within AWS, a common design pattern is to split a system into services that are responsible for processing customer requests (the data plane), and services that are responsible for managing

and vending customer configuration by, for example, creating, deleting and modifying resources (the control plane). Amazon EC2 is an example of an architecture that includes a data plane and a control plane. The data plane consists of EC2 physical servers where customers' EC2 instances run. The control plane consists of a number of services responsible for communicating with the data plane, and performing functions such as relaying commands to launch or terminate an instance or ingesting operational telemetry.



Nitro System control architecture

The Nitro Controller presents to the dedicated EC2 control plane network a set of strongly authenticated and encrypted networked APIs for system management. Every API action is logged and all attempts to call an API are cryptographically authenticated and authorized using a fine-grained access control model. Each control plane component is authorized only for the set of operations needed for it to complete its business purpose. Using formal methods, we've proven that the network-facing API of the control message parsing implementation of the Nitro Controller is free from memory safety errors in the face of any configuration file and any network input.

Nitro Cards for I/O

In addition to the Nitro Controller, some systems use additional specialized Nitro Cards to perform specific functions. These subordinate Nitro Cards share the same SoC and base firmware designs as the Nitro Controller. These Nitro Cards are designed with additional hardware and specialized firmware applications as required for their specific functions. These include, for example, the Nitro Card for VPC, the Nitro Card for EBS, and the Nitro Card for Local NVMe Storage.

These cards implement data encryption for networking and storage using hardware offload engines with secure key storage integrated in the SoC. These hardware engines provide encryption of both local NVMe storage and remote EBS volumes without practical impact on their performance. The last three versions of the Nitro Card for VPC, including those used on all newly released instance types, transparently encrypt all VPC traffic to other EC2 instances running on hosts also equipped with encryption-compatible Nitro Cards, without performance impact.

Note
AWS provides secure and private connectivity between EC2 instances of all types. In addition, some instance types use the offload capabilities of the underlying Nitro System hardware to automatically and transparently encrypt and anonymize in-transit traffic between instances, using AES-256-GCM. There is no impact on network performance. To support this additional in-transit traffic encryption between instances the instances must be supported instance types, in the same Region, and in the same VPC or peered VPCs. The traffic must not pass through

a virtual network device or service such as a load balancer or transit gateway. For additional information and a list of supported instance types, refer to Encryption in transit.

The encryption keys used for EBS, local instance storage, and for VPC networking are only ever present in plaintext in the protected volatile memory of the Nitro Cards; they are inaccessible to both AWS operators as well as any customer code running on the host system's main processors. Nitro Cards provide hardware programming interfaces over the PCIe connection to the main server processor—NVMe for block storage (EBS and instance store), Elastic Network Adapter (ENA) for networking, a serial port for out-of-band OS console logging and debugging, and so on.

Note
EC2 provides customers with access to instance console output for troubleshooting. The Nitro System also enables customers to connect to a serial console session for interactive troubleshooting of boot, network configuration, and other issues. "Out-of-band" in this context refers to the ability for customers to obtain information or interact with their instances through a channel which is separate from the instance itself or its network connection.

When a system is configured to use the Nitro Hypervisor, each PCIe function provided by a Nitro Card is sub-divided into virtual functions using single-root input/output virtualization (SR-IOV) technology. This facilitates assignment of hardware interfaces directly to VMs. Customer data (content that customers transfer to us for processing, storage, or hosting) moves directly between instances and these virtualized I/O devices provided by the Nitro Cards. This minimizes the set of software and hardware components involved in the I/O, resulting in lower costs, higher performance, and greater security.

The Nitro Security Chip

The Nitro Controller and other Nitro Cards together operate as one domain in a Nitro System and the system main board with its Intel, AMD, or Graviton processors, on which customer workloads run makes up the second. While the Nitro Controller and its secure boot process provide the hardware root of trust in a Nitro System, an additional component is used to extend that trust and control over the system main board. The Nitro Security Chip is the link between those two domains that extends the control of the Nitro Controller to the system main board, making it a subordinate component of the system, thus extending the Nitro Controller chain of trust to cover it. The following sections detail how the Nitro Controller and Nitro Security Chip function together to achieve this goal.

The Nitro Security Chip protection of system hardware

The Nitro Security Chip is a device integrated into the server's system main board. At runtime, it intercepts and moderates all operations to local non-volatile storage devices and low speed system management bus interfaces (such as Serial Peripheral Interface (SPI) and I^2C)—in other words, to all firmware.

The Nitro Security Chip is also situated between the BMC and the main system CPU on its high-speed PCI connection, which enables this interface to be logically firewalled on production systems. The Nitro Security Chip is controlled by the Nitro Controller. The Nitro Controller, through its control of the Nitro Security Chip, is therefore able to mediate and validate updates to the firmware, or programing of other non-volatile devices, on the system mainboard or Nitro Cards of a system.

A common industry practice is to rely on a hypervisor to protect these system hardware assets, but when no hypervisor is present—such as when EC2 is used in bare metal mode—an alternative mechanism is required to ensure that users cannot manipulate system firmware. The Nitro Security Chip provides the critical security function of ensuring that the main system CPUs cannot update firmware in bare metal mode. Moreover, when the Nitro System is running with the Nitro Hypervisor, the Nitro Security Chip also provides defense in depth in addition to the protections for system hardware assets provided by the hypervisor.

The Nitro Security Chip at system boot or reset

The Nitro Security Chip also provides a second critical security function during system boot or reset. It controls the physical reset pins of the server system main board, including its CPUs and the BMC, if present. This enables the Nitro Controller to complete its own secure boot process, then verify the integrity of the basic input/output system (BIOS), BMC, and all other system firmware before instructing the Nitro Security Chip to release the main CPUs and BMC from being held in reset. Only then can the CPUs and BMC begin their boot process using the BIOS and firmware that have just been validated by the Nitro Controller.

The Nitro Hypervisor

The Nitro Hypervisor is a limited and carefully designed component that has been intentionally minimized and purpose built with the capabilities needed to perform its assigned functions, and no more. The Nitro Hypervisor is designed to receive virtual machine management commands (start, stop, and so on) sent from the Nitro Controller, to partition memory and CPU resources by utilizing hardware virtualization features of the server processor, and to assign SR-IOV virtual functions provided by Nitro hardware interfaces (NVMe block storage for EBS and instance storage, Elastic Network Adapter (ENA) for network, and so on) through PCIe to the appropriate VM.

Note
While the Nitro architecture is unique in that it does not require the use of a hypervisor to provide software-defined infrastructure, in most customer scenarios, virtualization is valuable because it allows very large servers to be subdivided for simultaneous use as multiple instances (VMs) as well as other advantages such as faster provisioning. Hypervisors are required in virtualized configurations to provide isolation, scheduling, and management of the guests and the system. Therefore, the Nitro Hypervisor plays a critical role in securing a Nitro-based EC2 server in those common scenarios.

Some instance types built on the Nitro System include hardware accelerators, both built by AWS and by third parties (such as graphics processing units, or GPUs). The Nitro Hypervisor is also responsible for assigning these hardware devices to VM, recovering from hardware errors, and performing other functions that cannot be performed through an out-of-band management interface.

Within the Nitro Hypervisor, there is, by design, no networking stack, no general-purpose file system implementations, and no peripheral device driver support. The Nitro Hypervisor has been designed to include only those services and features which are strictly necessary for its task; it is not a general-purpose system and includes neither a shell nor any type of interactive access mode. The small size and relative simplicity of the Nitro Hypervisor is itself a significant security benefit compared to conventional hypervisors.

The Nitro Hypervisor code is a managed, cryptographically signed firmware-like component stored on the encrypted local storage attached to the Nitro Controller, and is therefore chained to the hardware root of trust of the Nitro Controller. When a system is configured to use the Nitro Hypervisor, the Nitro Controller directly loads a known-good copy of the hypervisor code onto the system main board in a manner similar to firmware.

Note
The mechanism for this hypervisor injection process is the use of a read-only NVMe device provided by the Nitro Controller to the main board as the system *boot drive.*

Offloading data processing and I/O virtualization to discrete hardware, and reducing the set of responsibilities of the hypervisor running on the host CPU, is central to the Nitro System design. It not only provides improved performance and stronger security through isolation, it also enables the bare metal instance type feature of EC2 because the hypervisor is now an optional discrete component that is no longer needed in order to provide I/O virtualization, system management, or monitoring.

> **Note**
> The Nitro System enables what is effectively "bare metal" performance by running nearly all the functionality of the virtualization system on the Nitro cards rather than on the host's system mainboard CPUs. Refer to Bare metal performance with the AWS Nitro System.

The meticulous exclusion of non-essential features from the Nitro Hypervisor eliminates entire classes of bugs that other hypervisors can suffer from, such as remote networking attacks or driver-based privilege escalations. Even in the unlikely event of a bug being present in the Nitro Hypervisor that allows access to privileged code, it still presents an inhospitable environment to any potential attacker due to the lack of standard operating system features such as interactive shells, filesystems, common user space utilities, or access to resources that could facilitate lateral movement within the larger infrastructure.

For example, as previously noted, the Nitro Hypervisor has no networking stack and no access to the EC2 network. Instead, the Nitro Controller and other Nitro Cards mediate all access of any kind to the outside network, whether the main server processor is running the Nitro Hypervisor or running in bare metal mode. Moreover, as discussed in detail next, the passive communications design of Nitro means any attempted "outreach" from code running in the context of the hypervisor to the Nitro Cards will be denied and alarmed.

Update process for the Nitro Hypervisor

A crucial aspect of maintaining system security is routine updates. The Nitro Hypervisor allows for full system live update. When a new version of the Nitro Hypervisor is available, the full running hypervisor code is replaced in-place while preserving customer EC2 instances with near-imperceptible performance impact to those instances. These update processes are designed such that at no time do any of the security protocols or defenses of the Nitro System need to be dropped or relaxed. This overall live update capability is designed to provide zero downtime for customers' instances, while ensuring that not only new features but also security updates can be applied regularly and rapidly.

The decoupling of customer instance downtime from Nitro System component updates eliminates the need for the EC2 service to carefully balance tradeoffs between customer experience and potential security impact when planning system updates, thereby yielding improved security outcomes.

Putting the pieces together: EBS volume attachment

To create a better picture of how many of the pieces of the Nitro System operate together, let's review what happens when a customer makes a public EC2 API call that changes the running state of their EC2 instance on a Nitro System. In particular, we'll look at the case where a customer attaches an existing encrypted EBS volume to a running instance.

In the first step, the customer uses the AWS Command Line Interface (AWS CLI), AWS SDK, or AWS Management Console to invoke the `AttachVolume` command, targeting the instance. After validating that the customer's IAM identity is authenticated and authorized to make the `AttachVolume` command, the API call is processed by a set of microservices inside the EC2 and EBS control planes. In the end, the control plane services call a defined set of encrypted, authenticated network APIs provided by the Nitro Controller with the information required to allocate the resources needed to attach the volume. Multiple services are involved in this operation, and each microservice has separated duties that limit the scope of access to the Nitro Controller APIs.

The EC2 control plane allocates the PCIe device resources of the Nitro Card for EBS that are required to service reads and writes to the logical EBS volume (either a NVMe virtual function for a virtualized instance or NVMe physical function for a bare metal instance). The EBS control plane provides the information needed to connect to the EBS servers that house the encrypted volume data over the network, and also an encrypted copy of the volume's data key that is stored as volume metadata. The encrypted data key is protected by an AWS KMS key present only inside the AWS Key Management Service (AWS KMS); therefore, as part of the process for attaching the volume, the encrypted key must be sent to AWS KMS for decryption.

Assuming that the customer IAM identity that made the `AttachVolume` command is also authorized to make a Decrypt command in AWS KMS under the expected AWS KMS key, the encrypted volume's data key will be decrypted. The Nitro System's access to this operation is protected by AWS KMS Grants and by IAM Forward Access Sessions. (Refer to this explanation of IAM Forward Access Sessions in the context of Elastic Load Balancing, AWS Certificate Manager and AWS Key Management Service in a presentation by Colm MacCárthaigh, VP and Distinguished Engineer at AWS.)

Together, these mechanisms cryptographically ensure that the Nitro System is granted access to use a customer's AWS KMS managed key only when the customer has recently authorized and authenticated this access. The Nitro System is not granted use of AWS KMS-managed keys on an ad-hoc basis or absent a recent customer authorization.

After being decrypted inside AWS KMS, and before being sent to the Nitro Controller using an encrypted Transport Layer Security (TLS) network connection, AWS KMS encrypts the data key again using a public key that serves as the cryptographic digital identity for the individual production Nitro host server. This public key was sent along with the data key for the encrypted volume by the EBS control plane to AWS KMS. Therefore, in addition to the entire message being encrypted on the wire by TLS, the data key is also asymmetrically encrypted within the message—double-encrypted on the wire. Only the Nitro Card of that specific production host with the specific customer's compute environment has the private key necessary to decrypt the encrypted data key. Once locally decrypted, that plaintext data key is stored only in volatile memory on that single Nitro Card during the time the volume is attached and in use.

Now the Nitro EBS Card is ready to present the EBS volume to the EC2 instance through a PCIe attachment of an NVMe interface. When the host is configured to use the Nitro Hypervisor, the Nitro Controller sends a message over the PCIe interface to instruct the Nitro Hypervisor to assign the NVMe virtual function for that EBS volume to the appropriate EC2 instance. The hypervisor then sends a virtual hardware hot-plug event to the VM to alert the customer-provided system software that a new NVMe block device is available. In the case of a bare-metal instance, the Nitro Card for EBS signals a PCIe hot plug event directly to the server processor, and the customer-provided system software running on the processor handles the PCIe hot-plug event of the NVMe device as it would on any other server.

At this point, the customer instance operating system running either as a virtualized guest or a bare metal instance interacts with a NVMe device presented by the Nitro Card for EBS over the PCIe interface. This interaction occurs either as an SR-IOV function in the case of virtualized EC2 instances, or a PCIe physical function in the case of bare metal EC2 instances. The NVMe commands sent over the PCIe interface are handled by firmware running on the Nitro Card for EBS, which in turn interacts with the EBS service via the Nitro SoC's integrated network interface. And, as previously noted, the Nitro EBS Card is also able to offload the cryptographic operations for AES-256 XTS-encrypted EBS volumes so that every single block of customer data is fully encrypted before leaving the Nitro Card with no performance impact. A customer may also choose to use an encrypting filesystem at the operating system level so that all customer data is fully encrypted before it is written or transmitted to the Nitro Card for EBS. This approach also establishes an additional layer of encryption for EBS data both on the wire and in the EBS storage system.

No AWS operator access

By design the Nitro System has no operator access. There is no mechanism for any system or person to log in to EC2 Nitro hosts, access the memory of EC2 instances, or access any customer data stored on local encrypted instance storage or remote encrypted EBS volumes. If any AWS operator, including those with the highest privileges, needs to do maintenance work on an EC2 server, they can only use a limited set of authenticated, authorized, logged, and audited administrative APIs. None of these APIs provide an operator the ability to access customer data on the EC2 server. Because these are designed and tested technical restrictions built into the Nitro System itself, no AWS operator can bypass these controls and protections.

As with most engineering decisions, the choice to design the Nitro System without a mechanism for operator access came with trade-offs. In rare cases subtle issues can arise that, because there are no general access capabilities on our production hardware, AWS operators are unable to debug in-place. In those rare circumstances we must work with customers, at their request, to reproduce those subtle issues on dedicated non-production Nitro debugging hardware. This can be less convenient than if our operators could debug in-place, but we strongly believe that this is the better trade-off for our customers. As a result, we must by necessity hold ourselves to the highest standard for system quality and testing prior to production release.

Passive communications design

The AWS Nitro System follows a “passive communication” design principle. What that means is that during production operation, components of the system never initiate outbound communication including to any control plane, management, or cloud service. Instead, there is a single hardened trusted service listening on the network, they listen for commands on the network or system bus, take action based on those commands, and then return results—all through well-defined APIs with scoped-down access controls. Both sides of these communication paths also perform parameter validation to ensure that only valid parameters are sent and received.

This pattern begins with the hypervisor itself. It waits for commands from the Nitro Controller on a private channel over PCIe. It never initiates outbound communication with the rest of the Nitro System. It *cannot* initiate any outbound network connections because, as noted, it has no networking stack. If at any time via an unlikely series of actions, the Nitro Hypervisor should attempt to initiate communications with other Nitro System components, this occurrence would be a clear signal of a firmware flaw or possible system compromise, and the EC2 service is designed to react accordingly to prevent impact and alarm for operator response.

Note
In bare metal mode, there is no hypervisor running on the server processor to await instructions from the Nitro Controller to start, stop, or reset the host server. In that case, the Nitro Controller controls the main board through its private BMC connection and the Nitro Security Chip.

The passive communications model repeats at the next layer out in the Nitro System. The Nitro Controller listens on a secure network channel awaiting authenticated and authorized commands in the form of specific APIs invoked by the EC2 control plane. The Nitro Controller never initiates any outbound communications on the EC2 control plane network. Even logical “push” features such as publishing CloudWatch metrics for the EC2 instances running on the host, or sending off the Nitro API logs to the EC2 control plane are implemented as a “pull” process. The control plane polls the Nitro Controller periodically to retrieve the metrics using well-defined APIs. Any attempt at outbound communication from the Nitro Controller would be clear signal of a firmware bug or possible system compromise, for which the EC2 service is designed to react accordingly to prevent impact and alarm for operator response.

The net effect of the passive communications model is a high degree of isolation and safety. Because normal operation involves listening only for well-defined, parameter-validated messages and responding to them using well-defined, parameter-validated responses, the system is designed to clearly identify and alert on potential aberrant activity. The system is designed such that, in the unlikely event of a firmware bug on the system main board, it would be highly likely that a potential adversary attempting to escape from the system mainboard outwards to the Nitro Cards would be detected, blocked, and alarmed. Moreover, even in the extremely unlikely case that a potential adversary in the former case were able to escape the system mainboard and somehow gain access to the Nitro Cards, the Nitro System design again makes it highly likely that any attempt by that adversary to escape from the Nitro Cards would be detected, blocked, and alarmed for the very same reason. These multiple layers of defense protect not only the EC2 service itself, but also all customers running workloads inside the EC2 system.

Change management for the Nitro System

The software and firmware underlying the Nitro System is developed by diverse and globally distributed teams of engineers. All Nitro System related configurations and code changes are subject to multi-party review and approval, and staged rollouts in both testing and production environments. The software development starts with design documents and reviews, and moves through code reviews. A security review will be conducted by both the independent AWS Security team as well as the Amazon EC2 engineering team for significant changes or features.

All changes are reviewed by at least one additional engineering team member or stakeholder, as well as by an engineer with substantial EC2 tenure serving as a member of our Change Management Bar Raiser program. In addition to expert human review, all code check-ins must pass a battery of automated quality and security checks that cannot be by-passed and that run automatically under the control of a central build service which ensures deployment best practices are followed, including proper monitoring and rollback.

Once code reviews and approvals are complete, and all automated checks are passed, our automated package deployment process takes over. As part of this automated deployment pipeline, binary artifacts are built and teams run end-to-end, validation, and security-specific tests. If any type of validation fails, the deployment process is halted until the issue is remediated.

Software and firmware binaries are cryptographically signed using an asymmetric private key which is only accessible through the automated pipeline, which logs all key signing activity.

Signed software and firmware are deployed to the EC2 fleet by a dedicated EC2 deployment system, which is configured to follow a defined deployment policy and schedule. Changes roll out in waves across Availability Zones and Regions. Deployments are monitored to ensure that only software versions that function as intended remain deployed and that any anomalous behavior triggers automatic rollbacks.

Note
Refer to the Amazon Builder's Library for more on how Amazon builds and operates software. Specifically, Automating safe, hands-off deployments by Clare Liguori, Sr. Principal Engineer at AWS, Going faster with continuous delivery by Mark Mansour, Senior Manager of Software Development at AWS, and Ensuring rollback safety during deployments by Sandeep Pokkunuri, Senior Principal Engineer at AWS.

The EC2 approach to preventing side-channels

Since its inception, EC2 has consistently taken a conservative approach to designing and operating secure multi-tenant systems for our customers. Our design approach favors simple and reliable abstractions, which provide strong isolation between security domains and limit the sharing of critical system resources across customers. AWS designs our systems to not only provide defense-in-depth against known security threats, but also to avoid impact from classes of potential security issues which do not have known practical exploitation techniques. In addition to the thoroughly tested and well-established security mechanisms we employ in production, AWS is actively engaged with the cutting edge of security research to ensure that we remain not only up-to-date, but are actively looking around corners for security issues on behalf of our customers.

Research and disclosures in the area of microarchitectural side-channels published over the past few years have brought concerns around this topic to the forefront. Side channels are mechanisms that potentially allow revealing secret information in a computer system through the analysis of indirect data gathered from that system. An example of such indirect data may be the amount of time it takes for a system to operate on an input. In some cases, although a system never directly reveals a secret piece of data, an external party may be able to determine the value of that secret through careful analysis of differences in time taken to process carefully selected inputs

Note
A simple example of such a scenario would be a program which receives a password in the form of a string as an input and validates whether that string matches the secret value. This program analyses the provided string one character at a time comparing each character to the corresponding character of the secret and returns an error as soon as it encounters a mismatch. Although the program never provides the requester with the value of the secret, the program "leaks" information about the secret in the form of a different response time for an input that starts with one or more of the same characters as the secret as for one which does not. Through a process of systematic trial and error an observer may be able to measure the time taken to respond to certain inputs in order to determine the value of the secret one character at a time.

Careful deployment of countermeasures such as those employed by s2n-tls, the open-source SSL/TLS implementation from AWS, can be used to protect against these forms of side-channel data disclosure.

Note
s2n-tls incorporates and proves using formal methods time-balancing countermeasures to ensure that process timing is negligibly influenced by secrets, and therefore no attacker-observable timing behavior depends on secrets. For more on these countermeasures in s2n-tls and the formal proof of those countermeasures, refer to SideTrail: Verifying Time-Balancing of Cryptosystems.

Microarchitectural side-channels specifically involve the manipulation of the low-level behavior of a system's processor in certain circumstances, to allow one process running on that system to indirectly ascertain the value of secret data through system resources such as caches, internal buffers, and other stateful data sources which it is not permitted to access directly. Critically, these side-channels center around the sharing of access to low-level hardware resources between two systems.

AWS has a conservative approach to EC2 tenant-isolation, discussed in the sections that follow, that is designed so that customer instances can never share system resources such as L1/L2 cache or threads running on the same CPU core. This fundamental design choice rules out the possibility of data leakage from customer instances through microarchitectural side-channels which are predicated upon shared access to these resources among tenants.

Side-channel protections in the broader EC2 service

All EC2 instances include robust protections against side-channels. This includes both instances based on the Nitro System or on the Xen hypervisor. While this section discusses these protections in terms of the Nitro System, these protections are also present in Xen-based EC2 instances.

Virtualized EC2 instance types fall into two categories:

- **Fixed performance instances**, in which CPU and memory resources are pre-allocated and dedicated to a virtualized instance for the lifetime of that instance on the host and
- **Burstable performance instances**, in which CPU and memory resources can be overcommitted in order to support larger numbers of virtualized instances running on a server and in turn offer customers a reduced relative instance cost for applications with low-to-moderate CPU utilization. Refer to Burstable performance instances.

In either case, the design and implementation of the Nitro Hypervisor includes multiple protections for potential side channels.

For fixed performance instances, dedicating resources provides both natural protection against side channels and higher performance compared to other hypervisors. For example, a c5.4xlarge instance is allocated 16 virtual CPUs (eight cores, with each core providing two threads) along with 32 GiB of memory. When an instance is launched, the EC2 control plane instructs the Nitro Controller to allocate the necessary CPU, memory, and I/O resources to support the instance.

The Nitro Hypervisor is directed by the Nitro Controller to allocate the full complement of physical cores and memory for the instance. These hardware resources are “pinned” to that particular instance. The CPU cores are not used to run other customer workloads, nor are any instance memory pages shared in any fashion across instances— unlike many hypervisors that can consolidate duplicated data and/or instruction pages to conserve physical memory.

Even on very small Nitro EC2 instances with limited resources, CPU cores are never simultaneously shared between two customer instances through Simultaneous Multi-Threading (SMT). Customer instances are provided with multiples of either two vCPUs, representing two threads of a single core for processors employing SMT, or a single vCPU for processor configurations that use exclusive full-core threading (such as the AWS Graviton processors). No sharing of cores means that no Level 1 or Level 2 caches or other core-specific resources such as speculative execution or power-saving state are shared.

Some instance sizes can share some last level cache lines non-simultaneously. Although it is possible to use priming and probing of last level cache lines as an ultra-low bandwidth signal between witting co-operating processes, this is not the same as a practical side-channel. By virtue of its function, only relatively infrequently accessed data is referenced in last-level cache lines. Side-channels typically require a very large and statistically relevant number of samples in order to over-come the noise present in systems.

No practical attack has been demonstrated when, as with EC2, memory pages are not share across virtual. All practical microarchitectural side-channel attacks to date have used either shared cores via SMT or shared L1/L2 caches, or other low-level core attributes such as floating point units. Side-channel attack mitigations are very strong in EC2 because those resources are never shared in an EC2 Nitro environment.

Note
EC2 accurately exposes the underlying CPU topology of the hardware, including last-level (typically L3) cache and non-uniform memory access (NUMA) information, directly through to instances. It is therefore possible for customers to determine by inspection what size instance is allocated the number of CPU cores needed to “fill” exactly one or more than one of the CPU

segments which share an L3 cache, and thus determine whether or not a given instance shares any L3 cache with another instance. L3 cache sharing topologies differ between CPU designs, and may be shared across a core, CPU complex, or Core complex die depending on the processor architecture. For example, in a typical two-socket Intel-based EC2 system, an instance size that is one-half the largest size will fill a CPU core and will not share the L3 cache with another instance.

As previously mentioned, burstable performance EC2 instances (for example, T3, T3a, and T4g) can utilize overcommitted CPU and memory resources. The CPU resources needed to run burstable performance instances are scheduled according to a credit-based allocation. In that low cost but relatively high-performance family of instances, even the smallest instance types still provide customers with a minimum of two vCPUs (one core, two threads) on processors that utilize SMT.

It is possible, however, for two burstable performance EC2 instances to run *sequentially* (not simultaneously) on the same core. It is also possible for physical memory pages to be reused, remapped, and swapped in and out as virtual memory pages. However, even burstable instances never share the same core at the same time, and virtual memory pages are never shared across instances.

The Nitro Hypervisor utilizes a number of safety strategies at each context switch between instances to ensure that all state from the previous instance is removed prior to running another instance on the same core(s). This practice provides strong mitigation against possible side-channel attacks.

For burstable performance EC2 instances, the Nitro System may employ memory management techniques such as reusing, remapping or swapping physical memory out as virtual memory pages but the system is designed so that virtual memory pages are never shared across instances in the interest of maintaining a strong isolation boundary.

Finally, burstable performance instances—whether those being targeted or those seeking to detect data through side-channel techniques—may be rescheduled on different cores than previously used, further limiting the possibility of any kind of successful timing-based security issue.

Additional side-channel benefits of the Nitro System

In addition to the protections provided by EC2 for both Xen and Nitro, there are some non-obvious but very important benefits in the design of the Nitro System and the Nitro Hypervisor when it comes to side-channel concerns. While, for example, some hypervisors required extensive changes to implement address space isolation as part of the mitigations for the L1 Terminal Fault transient execution side channel attack (for example, refer to Hyper-V HyperClear Mitigation for L1 Terminal Fault), the design and implementation of the Nitro System provided natural immunity, because the Nitro Hypervisor's virtual address space is isolated from memory allocated to customer instances.

We have also applied what we learned from designing the Nitro System to mitigate emerging threats of microarchitectural side channel attacks in the community version of the Xen hypervisor. Refer to Running Xen Without a Direct Map.

As discussed previously, the Nitro System dramatically decreases the amount of EC2 system code running on the main server processor itself which dramatically narrows the attack surface of the hypervisor and isolates customer I/O data processing from the rest of the system. The AWS code needed to provide the software-defined I/O features of EC2 does not run on the same processors that run customer workloads.

This compartmentalization and use of dedicated hardware means that customer data processing in I/O subsystems is isolated at the hardware function level, and does not reside in host memory, processor caches, or other internal buffers—unlike general-purpose virtualization software that does mix this data as a side effect of running on the shared host CPUs.

Underpinning all of these protections is that AWS is at the fore-front of security research and often leads the research and discovery of industry-impacting issues as well as the mitigation and coordination of issues.

Nitro Enclaves

Nitro Enclaves is a feature of the Nitro System that allows customers to divide their workloads into separate components that need not fully trust each other, as well as a means by which to run highly trusted code and process data to which the customer's EC2 instance administrators have no access. Its features and benefits are not covered in this paper, but the following is worth noting in this context.

A Nitro Enclave inherits the same isolation and side-channel mitigations as every other EC2 instance running on the same server processor. The parent instance must allocate a fixed number of vCPUs (the minimum amount equaling one full core) as well as a fixed number of memory pages. That fixed set of CPU and memory resources are subtracted from the parent instance (using the "hot-unplug of hardware resources" feature supported in both Linux and Windows kernels) and then utilized by the Nitro Hypervisor to create another fully protected independent VM environment in which to run the Nitro Enclave image.

All of the protections discussed above are automatically in place when using Nitro Enclaves since there is no core or memory sharing with the parent instance.

Closing thoughts on side channels

In summary, careful design decisions in Nitro and the EC2 platform provide a number of very strong mitigations against the possibility of practical side-channel attacks, including removing shared access between instances to the CPU and memory resources which these attacks require. Additionally, customers can optionally choose not to have their instances provisioned on the same hosts as instances belonging to other customers. Moreover, should any future research uncover new challenges, AWS participation in coordinated vulnerability response groups for Linux, KVM, Xen, and other key technologies as well as the Nitro System's live-update technologies design will allow AWS to react quickly to protect customers from new threats that emerge without disrupting customer workloads. AWS was a member of the small group of companies that worked on Spectre and Meltdown prior to public disclosure, and mitigated all risks in its infrastructure before the public disclosure.

> **Note**
> Customers may opt out of sharing compute hardware with other customers by using either the "Dedicated Instances" or the "Dedicated Hosts" features of EC2. These features represent instance placement strategies that result in a single customer being the only customer at any given time with instances scheduled on a particular EC2 physical host. Refer to Amazon EC2 Dedicated Hosts.

We continue to work with key partners such as Intel, AMD, and ARM on hardware security research and coordinated vulnerability response and continue to raise the bar with additional innovation for compute isolation. One such example is the open-source Firecracker VMM, which enables serverless container and function-based services such as AWS Fargate and AWS Lambda to benefit from the security, isolation, and consistency of virtualization without compromise on the speed, flexibility, and performance that customers require for these workloads.

> **Note**
> Firecracker is a virtualization technology that is purpose-built for creating and managing secure multi-tenant container and function-based services. Firecracker is a virtual machine monitor which manages workloads in lightweight microVMs. It implements a minimal device model that excludes all non-essential functionality and reduces the attack surface area of the microVM. In addition to the security and isolation-optimizations it employs, Firecracker also enables fast

boot times—initiating user space or application code in as little as 125ms---and providing a low memory overhead of as little as 5 MiB per microVM.

Side channel issues are a constantly evolving area of research and resulting innovation and mitigation. We believe that relying on AWS with its deep expertise and continuing focus on this topic is a good place for customers to place their bets when it comes to protection from future risks.

Note

Refer to this presentation on side channel issues by Eric Brandwine, VP and Distinguished Engineer at AWS. In the presentation he talks about the transition from Xen to Nitro (at 42.40) and the resulting advantages, but also importantly concludes by pointing out that this topic has become one like cryptography, where the most reasonable approach is to rely on deep experts and re-use their work (at 49.29).

Nitro System security in context

The Nitro System design features discussed in this paper operate in the context of the full set of robust controls in place at AWS to maintain security and data protection in the AWS Cloud. In this section we will provide a high-level overview of relevant AWS security and compliance practices. As an AWS customer you inherit all the best practices of AWS policies, architecture, and operational processes built to satisfy the requirements of our most security-sensitive customers.

AWS environments are continuously audited, with certifications from accreditation bodies across geographies and verticals. AWS Outposts also offers the ability, where required, for customers to run AWS compute, storage, database, and other services locally on Nitro System based hardware located in their own facilities.

Infrastructure security

Security at AWS starts with our core infrastructure—the hardware, software, networking, and facilities that run AWS Cloud services. Custom-built for the cloud and designed to meet the most stringent security requirements in the world, our infrastructure is monitored 24/7 to help ensure the confidentiality, integrity, and availability of your customer data. With AWS you can build on the most secure global infrastructure, knowing you always own your customer data, including the ability to encrypt it, move it, and manage retention.

Physical access

Physical access to AWS data centers is strictly controlled, both at the perimeter and at building entry points by professional security staff using video surveillance, two-factor and biometric authentication, intrusion detection systems, and other electronic means. Authorized staff must pass two-factor authentication a minimum of two times to access data center floors. All visitors are required to present identification and are signed in and continually escorted by authorized staff. AWS only provides data center access and information to employees and contractors who have a legitimate business need for such privileges.

When an employee no longer has a business need for these privileges, his or her access is immediately revoked, even if they continue to be an employee of Amazon or Amazon Web Services. All physical access to data centers by AWS employees is logged and audited routinely.

Media sanitization

Media storage devices used to store customer data are classified by AWS as Critical. AWS has exacting standards on how to install, service, and eventually destroy the devices when they are no longer useful. When a storage device has reached the end of its useful life, AWS decommissions media using techniques detailed in NIST 800-88. Media that stored customer data is not removed from AWS control until it has been securely decommissioned.

Data protection

All data flowing across the AWS global network that interconnects our data centers and Regions is automatically encrypted at the physical layer before it is transmitted between our secured facilities. Additional encryption layers exist as well; for example, all inter-Region VPC peering traffic, and customer or service-to-service TLS connections. We provide tools that allow you to easily encrypt your customer data in transit and at rest to help ensure that only authorized users can access it, using keys you control managed by AWS KMS, or managing your encryption keys with AWS CloudHSM using FIPS 140-2 Level 3 validated HSMs.

We also give you the control and visibility you need to help you comply with regional and local data privacy laws and regulations. The design of our global infrastructure allows you to choose Regions in which your customer data is physically located, helping you meet data residency requirements.

Conclusion

The AWS Nitro System offers a unique set of capabilities that allow it to support the most sensitive workloads in a multi-tenanted, hyper-scale cloud environment. These capabilities are based on the AWS investment in custom silicon and associated firmware in order to create a virtualization stack tuned specifically for this custom silicon. Since the beginning of 2018, all new Amazon EC2 instance types are based on the AWS Nitro System, providing customers with all the security and other benefits discussed in this paper. In light of these deep technology investments and the excellent AWS track record of workload isolation, customers can rely on AWS compute environments to provide excellent security for their most sensitive workloads.

Contributors

Contributors to this document include:

- J.D. Bean, Principal Security Architect, Amazon EC2
- Mark Ryland, Director, AWS Office of the CISO
- Matthew S. Wilson, Vice President / Distinguished Engineer, Amazon Web Services
- Colm MacCárthaigh, Vice President / Distinguished Engineer, Amazon Web Services
- Benjamin Serebrin, Principal Software Engineer, Amazon EC2

Document revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Initial publication (p. 27)	Whitepaper published.	November 18, 2022

Notices

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.