TECHNOLOGY GUIDE

Intel Corporation

Zero Trust – Rethink Zero Trust with Intel Confidential Computing

Authors

David Lu
Intel Corporation

Heqing Zhu
Intel Corporation

Tarun Viswanathan
Intel Corporation

Yiwen Li
Intel Corporation

John DiGiglio
Intel Corporation

Michihiro Koyama
Intel Corporation

Christof Fetzer
SCONTAIN UG

1 Introduction

Enterprises are moving towards a multi-cloud deployment model while adopting a zero-trust security model to ensure a consistent security posture in all cases. The right combination of hardware and software solutions is critical to realize the required security.

Intel launched the 3rd Gen Intel® Xeon® Scalable processor in 2021. This processor has multiple security features that can help significantly boost the security posture of a "Zero Trust" solution architecture, including Intel® Software Guard Extensions (Intel® SGX) facilitating confidential computing and Intel® Total Memory Encryption (Intel® TME) for memory encryption. Following the launch of the 3rd Gen Intel Xeon Scalable processor, all leading cloud service providers have launched instance types based on the 3rd Gen Intel Xeon Scalable processor.

Enterprise services often run in a hybrid and multi-cloud environment. From a solution perspective, it is critical to protect enterprise applications when working with confidential data such as username, password, database credentials, and API keys, when interacting with third-party services, credentials for service-oriented architecture communication, and more. Selecting the right software solution to implement Zero Trust is an important decision. The solution needs to be easy to use, industry proven, and support flexible deployment environments including on-premises and multi-public cloud.

This paper reviews the security features of the 3rd Gen Intel Xeon Scalable processor, such as Intel TME, cryptographic instruction performance enhancements, and Intel SGX, and demonstrates how to apply them with HashiCorp Vault to create a high performing Zero Trust solution. Enterprises can use this solution and associated collateral as a reference to replicate to other workloads as well.

This document is part of the Network Transformation Experience Kit, which is available at Network & Edge Platform Experience Kits.

Table of Contents

Figures

Tables

Document Revision History

REVISION	DATE	DESCRIPTION
001	April 2022	Initial release.
002	March 2023	Minor updates as per trademark and branding.

1.1 Terminology

Table 1. Terminology

ABBREVIATION	DESCRIPTION
AES	Advanced Encryption Standard
DCAP	Data Center Attestation Primitives
Enclave	Ring 3 application software running inside the Intel SGX protections
FW, UEFI FW	Firmware, Unified Extensible Firmware Interface FW
HSM	Hardware Security Module
ISA	Instruction Set Architecture
SGX	Software Guard Extensions
NIST	National Institute of Standards and Technology
OCI	Open Container Initiative
RSA	Rivest-Shamir-Adelman crypto algorithm
SCONE	A Secure Container Environment
TME	Intel® Total Memory Encryption (Intel® TME)
VM	Virtual Machine
XTS	XEX Tweakable Block Cypher with Ciphertext Stealing

1.2 Reference Documentation

Table 2. Reference Documents

REFERENCE	SOURCE
Intel Xeon Scalable Platform Built for Most Sensitive Workloads	https://www.intc.com/news-events/press-releases/detail/1423/intel-xeon-scalable-platform-built-for-most-sensitive
Crypto Acceleration: Enabling a Path to the Future of Computing	https://newsroom.intel.com/articles/crypto-acceleration-enabling-path-future-computing
Golang	https://go.dev/
Go-devel patch	https://go-review.googlesource.com/c/go/+/334610/
HashiCorp Vault	https://www.Vaultproject.io/ https://medium.com/hashicorp-engineering/hashicorp-Vault-performance-benchmark-13d0ea7b703f
SCONE	https://scontain.com/index.html?lang=en
Graphene	https://github.com/oscarlab/graphene
Alibaba Inclavare Container	https://github.com/alibaba/inclavare-containers
Occlum	https://github.com/occlum/occlum
Intel SGX Programming Reference and SDK for Linux	https://software.intel.com/content/www/us/en/develop/articles/intel-sdm.html#combined https://download.01.org/intel-sgx/latest/linux-latest/docs/ https://github.com/intel/linux-sgx
National Institute of Standards and Technology FIPS Publication 197, Advanced Encryption Standard (AES)	https://csrc.nist.gov/publications/detail/fips/197/final
3rd Gen Intel® Xeon® Scalable Processor - Achieving 1 Tbps IPsec with Intel® Advanced Vector Extensions 512 (Intel® AVX-512) Technology Guide	https://networkbuilders.intel.com/solutionslibrary/3rd-generation-intel-xeon-scalable-processor-achieving-1-tbps-ipsec-with-intel-advanced-vector-extensions-512-technology-guide
Create Intel SGX VM in the Azure portal	https://docs.microsoft.com/en-us/azure/confidential-computing/quick-create-portal
Intel® Software Guard Extensions (Intel® SGX) – Key Management Reference Application (KMRA) on Intel® Xeon® Scalable Processors Technology Guide	https://networkbuilders.intel.com/solutionslibrary/intel-sgx-kmra-on-intel-xeon-processors-technology-guide
Intel® Software Guard Extensions (Intel® SGX) – Key Management Reference Application (KMRA) on Intel® Xeon® Scalable Processors User Guide	https://networkbuilders.intel.com/solutionslibrary/intel-sgx-kmra-on-intel-xeon-processors-user-guide

REFERENCE	SOURCE
Intel® Software Guard Extensions (Intel® SGX) – Securing Private Keys in an Encrypted Enclave for Your Service Mesh Demo	https://networkbuilders.intel.com/intel-software-guard-extensions-intel-sgx-securing-private-keys-in-an-encrypted-enclave-for-your-service-mesh-demo

2 Security Features of 3rd Gen Intel® Xeon® Scalable Processor

The 3rd Gen Intel Xeon Scalable processor introduces important new security features, including the following:

- Intel Total Memory Encryption (Intel TME)
- Cryptography NI (instruction set architecture (ISA) for AES and RSA computation)
- Intel Software Guard Extension (Intel SGX)

These features, described in detail in following sections, address data security at runtime, in transit, and at rest in a more secure and affordable way while also offering a more secured execution environment that together can help improve Zero Trust security posture significantly. The following table shows example servers.

Table 3. Example Servers

VENDOR	SERVER MODEL EXAMPLES
Dell EMC	PowerEdge R750, R650, R550, R450, R250, R350
Hewlett Packard Enterprise (HPE)	ProLiant DL360 and DL380, Synergy 480, HPE Apollo 2000
Lenovo	ThinkSystem SR650 V2, SR630 V2, ST650 V2, SN550 V2
Supermicro	X12

2.1 Intel® Total Memory Encryption (Intel® TME)

Intel TME can encrypt the entirety of physical memory of a physical server system. This capability typically is enabled in the very early stages of the boot process with a small setting in the UEFI/BIOS. After it is configured and locked, the CPU is responsible for encrypting all data into the system memory. Intel TME is based on the National Institute of Standards and Technology (NIST) standard AES-XTS algorithm with 128-bit or 256-bit keys, depending on the algorithm availability and selection. The encryption key used for Intel TME uses a hardware random number generator implemented in the Intel® CPU, and the key is not accessible by software or with external interfaces.

Firmware Impacts: Intel TME is configurable through a UEFI/BIOS setting. Most server systems based on the 3rd Gen Intel Xeon Scalable processor allow the user to turn Intel TME on or off.

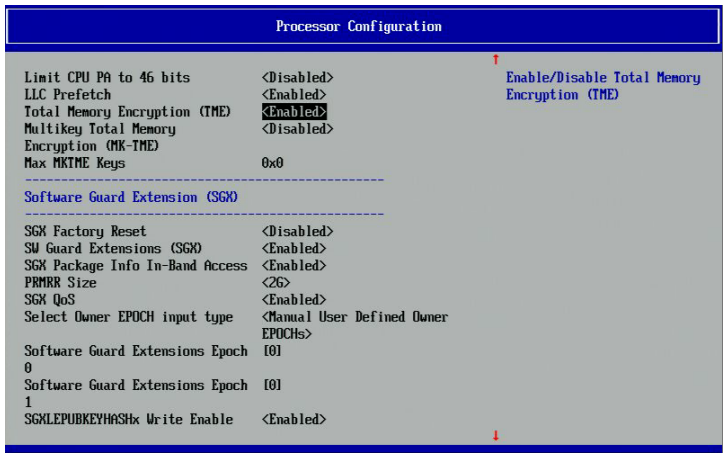


Figure 1. Processor Configuration with Total Memory Encryption Enabled

Software Impacts: Intel TME capability is transparent to software such as operating systems, hypervisors, or containers, applications, and micro services. It does not require any specific Linux kernel support. Overall, the performance impact of this capability is almost negligible when running software workloads such as Vault.

2.2 Encryption-Focused Instructions

Intel® Advanced Encryption Standard New Instructions (Intel® AES-NI) is a set of instructions available beginning with the Intel® Core™ processor family introduced in 2010. These instruction sets enable fast and more secure data encryption and decryption, using the Advanced Encryption Standard (AES) defined in NIST Federal Information Processing Standards (FIPS) publication number 197 (https://csrc.nist.gov/publications/detail/fips/197/final). Since AES is the dominant block cipher and is used in various protocols, the instructions are valuable for a wide range of applications. The architecture consists of six instructions that offer full hardware support for AES. Four instructions support AES encryption and decryption, and the other two instructions support AES key expansion. The AES instructions have the flexibility to support all uses of AES, including all standard key lengths, standard modes of operation, and even some nonstandard or future variants. They aim to offer a significant increase in performance compared to the current pure-software implementations.[1]

The additional instructions include VPMADD2 - vector instruction that does integer multiply accumulate, vAES - vector version of the Intel AES-NI instructions, vCLMUL - vector version of the CLMUL instruction, and Intel® Secure Hash Algorithm - New Instructions (Intel® SHA-NI). The combination of vAES and vCLMUL on wide registers that are available on the Intel® Advanced Vector Extensions 512 (Intel® AVX-512) further accelerate AES modes such as AES-CTR and AES-CBC. VPMADD2 is targeted at significantly reducing the instructions needed to generate public/private keys as part of an RSA-2K sign operation. Intel SHA-NI looks to improve hashing functions used in cryptographic protocols such as SSL/TLS as well helping with data deduplication in storage workloads.

Figure 2 illustrates the cryptographic algorithms' performance boost. The data was initially published at Hot Chips conference 2020.

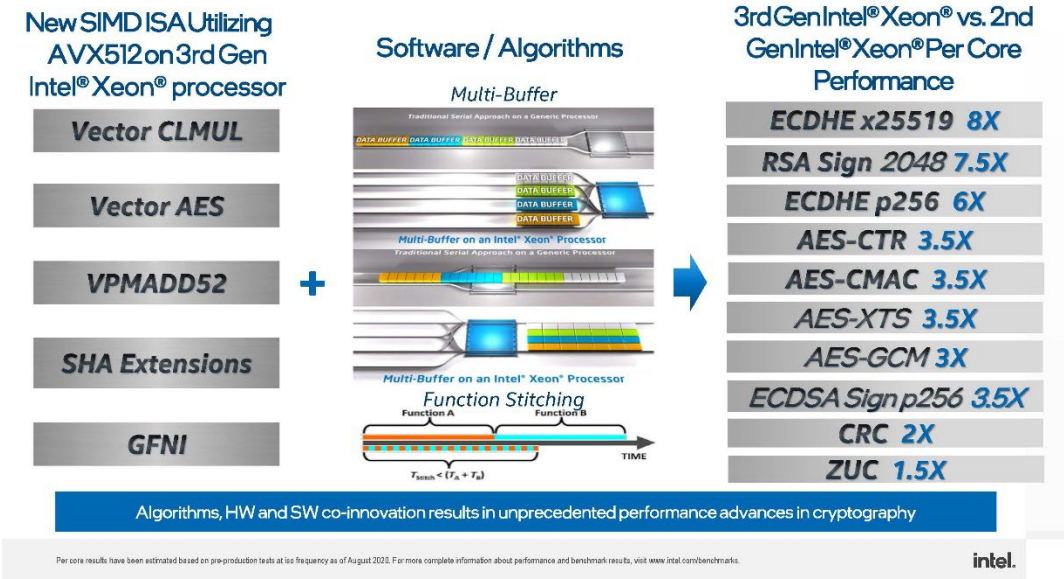


Figure 2. Performance Boost from Cryptographic Algorithms on 3rd Gen Intel Xeon Processor

Firmware Impacts: No impacts. Crypto ISA is transparent to UEFI/BIOS setting.

Software Impacts: To make the cryptographic-related instruction sets easy to use, Intel also developed a set of software libraries. Vault is written in golang, which is a very popular software programming language. Golang has a specific cryptographic package. Intel developed software patches and works with the community to upstream the code into the golang software community; the code patch is already in public domain. Follow the steps below to access the latest code patch.

```
# git clone https://go.googlesource.com/go
# cd go
# git fetch https://go.googlesource.com/go refs/changes/52/286852/1 && git checkout -b change-
286852 FETCH_HEAD
# cd src/
# ./all.bash
```

Intel also published a solution paper on how to achieve 1 Tbps IPsec with open source on a 3rd Gen Intel Xeon Scalable processor-based server system. The link to the paper is here.

[1] For workloads and configurations visit www.Intel.com/PerformanceIndex. Results may vary.

2.3 Intel® Software Guard Extensions (Intel® SGX)

Intel SGX is a set of instructions incorporated in the 3rd Gen Intel Xeon Scalable processor. Software developers can place security-sensitive codes and data into an Intel SGX enclave, which is then executed in a CPU protected region. Secure enclaves can be created on untrusted platforms not owned by the enterprise. Intel processor-based attestation can ensure the integrity of a secure enclave. After the enclave is verified, the remote attestation (application) can push secrets securely into the enclave. Even if the system is hosted in a third-party facility such as cloud, edge, or POP, the application can rely on Intel SGX to help secure the data and reduce the attacking surface available, such as to inside hackers or misconfiguration. The security natively available in the 3rd Gen Intel Xeon Scalable processor is enforced to create the trusted execution environment, which aligns with the Zero Trust principle. This alignment was not possible with previous generations of Intel Xeon Scalable processor. The following diagram illustrates that an operating system or hypervisor is not allowed to access application data.

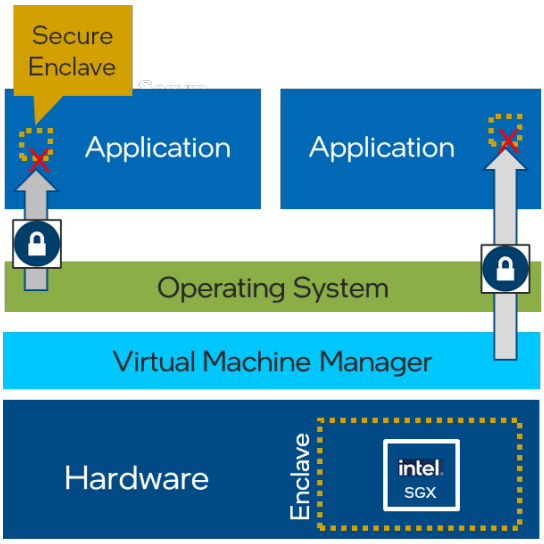


Figure 3. Intel SGX Security Model

Firmware Impacts: Server firmware must be Intel SGX-ready. This is supported by most 3rd Gen Intel Xeon Scalable processor-based server systems. You must turn on Intel SGX in the UEFI settings.

Software Impacts: Linux operating systems must use kernel 5.11 or a later version. Linux 5.11 includes the Intel SGX driver. There are two use cases, because Intel SGX is a set of new instructions.

1. Software applications can modify the code and use Intel SGX directly, for example, use Intel SGX for key management service. This gives one of the best security controls to the application developer.
2. You can run "unmodified software" in an Intel SGX enclave. Solutions from SCONE, Fortanix, and Anjuna can provide software tools that you can use to run an application within an Intel SGX enclave. In this paper, we provide a solution that uses HashiCorp Vault inside a SCONE container with Intel SGX on a 3rd Gen Intel Xeon Scalable processor. SCONE is a more secure container solution for Docker that uses Intel SGX trusted execution support to protect container processes from outside attacks.

There are open-source software solutions such as Graphene and Occlum that are also used to run an application in an Intel SGX enclave. Alibaba Inclavare Containers can enable a workload to run within an Open Container Initiative (OCI)-compliant container with Intel SGX-based protection.

3 Deployment Models

Here we introduce three "good – better – best" security models for Zero Trust – Vault deployment scenarios.

3.1 On-prem Deployment

Enterprise customers have control of the entire server.

- Good: Enable Intel TME to secure the entire memory.
- Better: Enable Vault with the advanced crypto capability, illustrated in Section 4.2.
- Best: Enable Vault with Intel SGX, using SCONE solution, illustrated in Section 6.

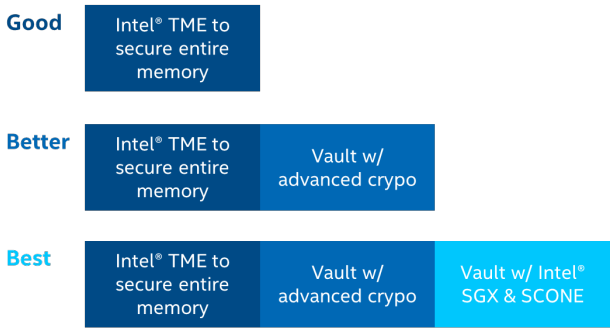


Figure 4. On-prem Deployment Good - Better - Best Models

3.2 Cloud Deployment

For cloud-based deployment, the cloud instance is the actual platform. Azure confidential computing VM and Alibaba confidential instances support Intel SGX-based technology. Intel TME is available in AWS m6i, c8i, and r6i instance types.

- Good: Select cloud instance based on a 3rd Gen Intel Xeon Scalable processor.
- Better: Enable Vault with the advanced crypto capability, illustrated in Section 4.2.
- Best: Select Azure/Alibaba confidential VM, deploy SCONE Vault, or open-source solution with confidential VM capability.

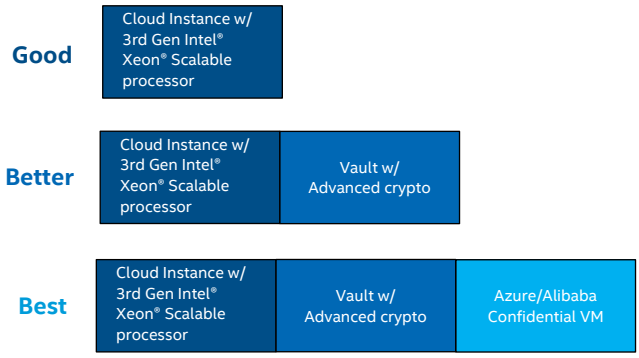


Figure 5. Cloud Deployment Good - Better - Best Models

After Vault is deployed into an Intel SGX enclave (protected area), it helps secure the most important data with confidential computing technology. One thing to consider is that large enterprises often include both legacy and modern software, and the legacy software is typically phased out over time. When selecting a Zero Trust solution provider, it is important to check if Vault integration is part of that solution. The solution architecture in Figure 6 illustrates the concept on a Zero Trust service mesh scenario. The numbers in the figure indicate where Intel SGX can be used.

- (1) Service mesh (Proxy) can use Vault to store secrets.
- (2) The data path engine can use Intel SGX for key management during the HTTPS handshake phase.

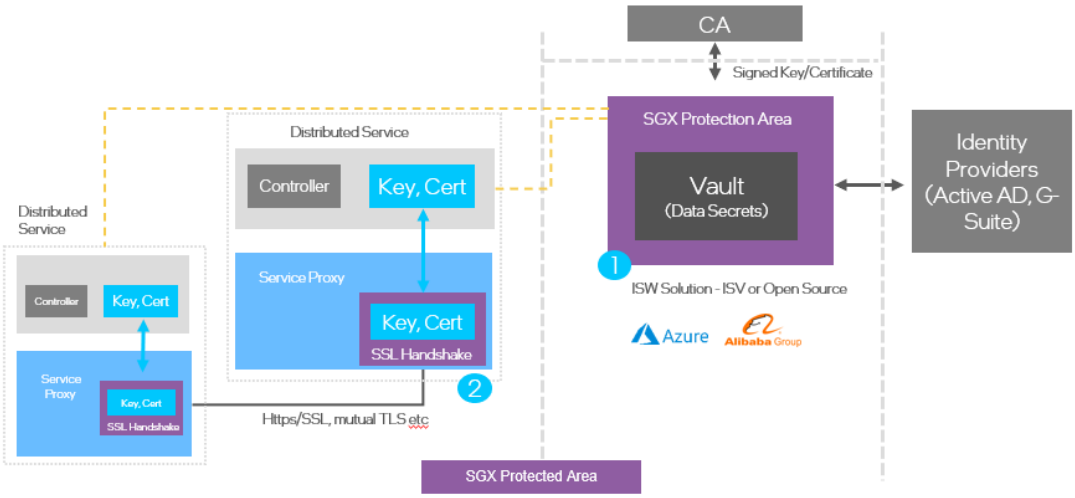


Figure 6. Zero Trust Service Mesh Architecture Example

4 Vault: Secret Manager Over Multi-Cloud

4.1 HashiCorp Vault

HashiCorp Vault is an industry-proven, multi-cloud-ready solution, available as open-source software. It is widely adopted in Zero Trust solutions to more securely access secrets consistently, in on-premises and multi-public cloud deployments. Vault can seal all important data secrets and store them externally, outside of application service. Vault is an important component for Zero Trust service mesh implementations usage, mTLS/HTTPS usage, and encrypted database usage. Vault also offers seamless integration with cloud and virtualized platforms, including Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform, VMware, and Red Hat. Compared to traditional hardware security module (HSM) appliances, Vault is easy to use in the cloud and cost effective. Increasing Vault's security and performance is highly desired in Zero Trust solutions.

- **Security**: Vault software runs on Linux servers. The SCONE confidential cloud-native computing solution from SCONTAIN UG can secure Vault with Intel SGX on a 3rd Gen Intel Xeon Scalable processor-based server or cloud instance (such as Azure confidential VM).
- **Performance**: As a centralized secret management service, Vault must address high load service demands. The 3rd Gen Intel Xeon Scalable processor adds cryptographic acceleration and can boost service capability to a new level.

A secret is anything that you want to tightly control access to, such as API keys, passwords, certificates, user names, and passwords. HashiCorp Vault is an open-source project and available at GitHub (https://github.com/hashicorp/vault). It is a well-known secret manager and provides a unified interface to any secret, while aiming at providing tight access control and recording a detailed audit log. Vault is a common architecture choice for enterprises to realize Zero Trust security solutions.

As a modern system requires access to a multitude of secrets: database credentials - API keys for external services, credentials for service-oriented architecture communication, and more - understanding who is accessing what secrets is already very difficult and platform specific. Adding on key rolling, secure storage, and detailed audit logs is almost impossible without a custom solution. This is where Vault steps in. The key features of Vault are:

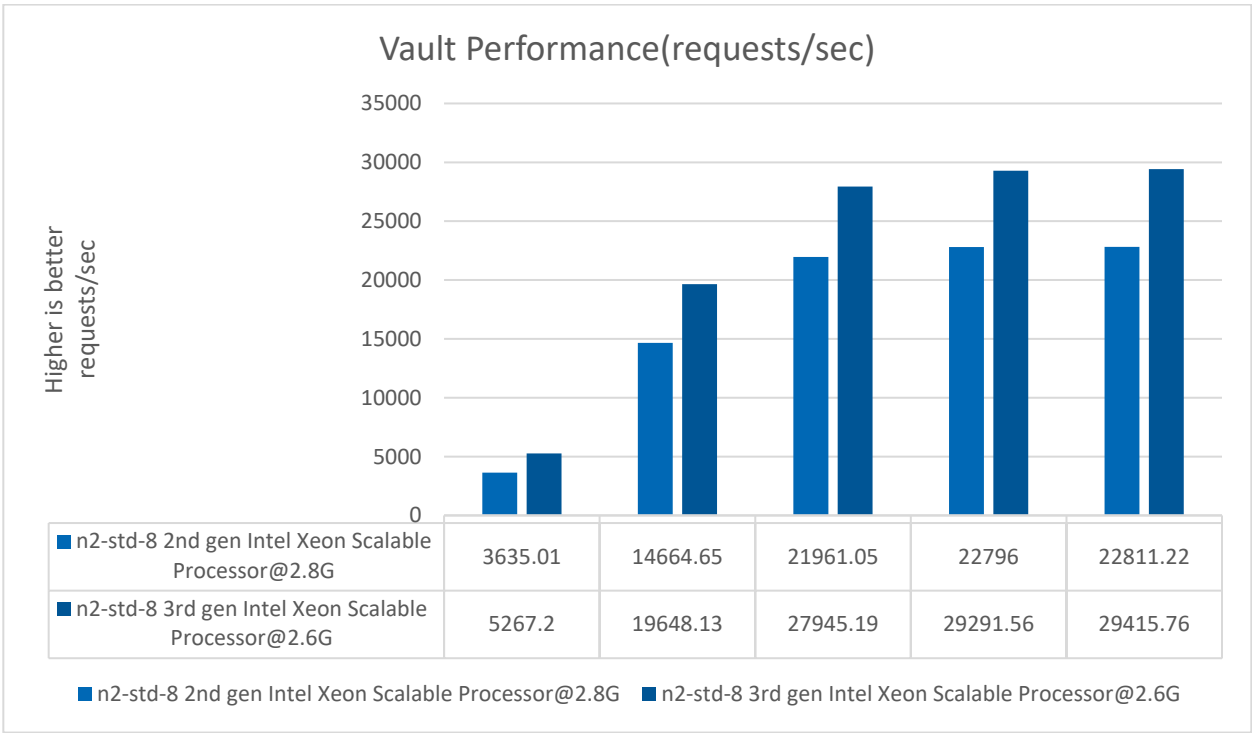
- Secure Secret Storage: Arbitrary key/value secrets can be stored in Vault. Vault is designed to encrypt these secrets before writing them to persistent storage, so gaining access to the raw storage is not enough to access your secrets. Vault can write to disk, Consul, and more.
- Dynamic Secrets: Vault can generate secrets on-demand for some systems, such as AWS or SQL databases. For example, when an application needs to access an AWS S3 bucket, it asks Vault for credentials. Vault generates an AWS keypair with valid permissions on demand. After creating these dynamic secrets, Vault also automatically revokes them after the lease is up.
- Data Encryption: Vault is designed to encrypt and decrypt data without storing it. This allows security teams to define encryption parameters and developers to store encrypted data in a location such as SQL without having to design their own encryption methods.
- Leasing and Renewal: Each secret in Vault has a lease associated with it. At the end of the lease, Vault automatically revokes that secret. Clients can renew leases via built-in renewal APIs.
- Revocation: Vault has built-in support for secret revocation. Vault can revoke not only a single secret, but also a tree of secrets, for example all secrets read by a specific user or all secrets of a particular type. Revocation assists in key rolling as well as locking down systems in the case of an intrusion.

However, Vault keeps all secrets in clear text in system memory, for example, just before they are sent to a client. In an extreme case, an attacker with root access has the possibility to access memory and may retrieve the critical secrets. Vault's encryption key to read from and to write to the external storage is also stored in main memory. Again, an attacker could gain access to the memory to reveal Vault's encryption key.

4.2 Vault Cryptographic Performance Boost from 3rd Gen Intel Xeon Scalable Processor

Vault is often used as encryption as a service. The cryptographic performance can be improved under Intel 3rd Gen Intel Xeon Scalable processor thanks for higher IPC and vAES. We tried to test the vault performance under Intel 2nd Gen Intel Xeon Scalable processor GCP instance and Intel 3rd Gen Intel Xeon Scalable processor GCP instance using HashiCorp's vault performance benchmark scripts (https://github.com/hashicorp/vault-guides/tree/master/operations/benchmarking/wrk-core-vault-operations) and the test method(https://medium.com/hashicorp-engineering/hashicorp-Vault-performance-benchmark-13d0ea7b703f) they provided.[2]

[2] See backup for workloads and configurations. Results may vary.



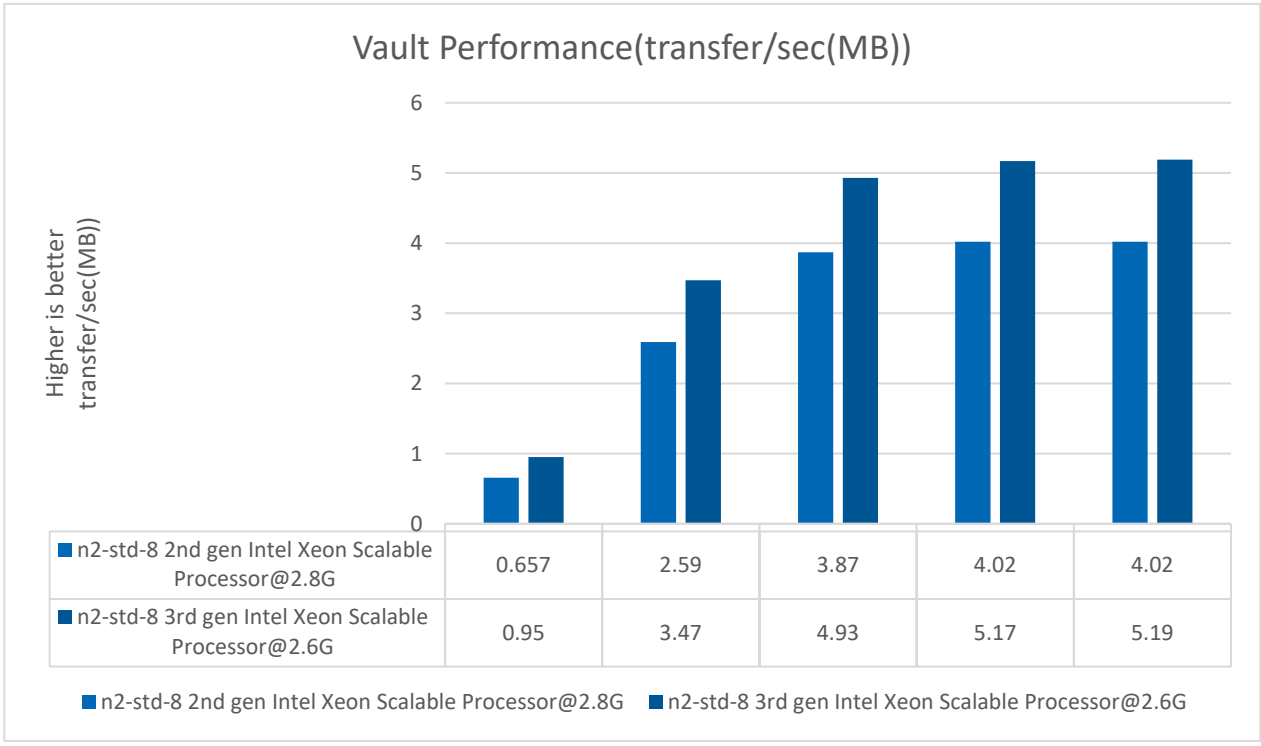


Figure 7. Vault Performance Results

As shown in Figure 7, our test results indicate both requests/sec and transfer/sec under n2-std-8 based on a 3rd Gen Intel Xeon Scalable processor GCP instance have up to 44% improved performance when compared with n2-std-8 based on a 2nd Gen Intel Xeon Scalable processor GCP instance.[3] See Appendix A for details on the hardware and software used in our testing.

[3] See backup for workloads and configurations. Results may vary.

5 Alibaba Open-Source Software for Intel SGX

Occlum is an open-source project (https://github.com/occlum/occlum) developed by Alibaba. It is a multi-process, memory-safe library operating system that enables applications run inside an enclave protected by Intel SGX. The unmodified Linux application can run in Intel SGX enclaves with three simple commands.

```
# occlum init
# occlum build
# occlum run
```

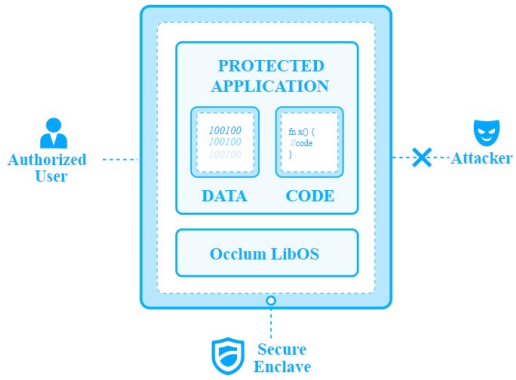


Figure 8. Occlum SGX High-Level Diagram

5.1 Run Vault Inside Occlum

Use the following steps to run Vault inside Occlum.

Step 1: Install the Intel SGX driver for Linux. The Intel SGX driver is included in Linux kernel 5.11 and later. If you are using an older Linux kernel version, install an out-of-tree driver.

Step 2: After the Intel SGX driver is installed, install the `enable_rdfsbase` kernel module, which enables Occlum to use rdfsbase-family instructions in enclaves. Detailed steps are:

```
# git clone https://github.com/occlum/enable_rdfsbase.git
# cd enable_rdfsbase && make && make install
```

Then, we can take the following steps to try out the Vault demonstration.

Step 1: If your Linux host machine does not have Docker, install the Docker engine. Then run the following Docker command to create the Occlum container.

```
# docker run -it  --device=/dev/sgx/enclave --device=/dev/sgx/provision occlum/occlum:0.22.0-ubuntu18.04
```

Step 2: Within the Occlum container, run the following commands to prepare the Vault running environment.

```
# git clone https://github.com/occlum/occlum.git
# cd occlum/demos/golang/Vault
# ./prepare_Vault.sh
```

Step 3: Start the Vault server.

```
# ./run_occlum_Vault_server.sh
```

Step 4: Test Vault by interacting with the Vault kv secret engine. Your result should be similar to the following.

```
# ./run_occlum_Vault_test.sh
./source_code/bin/Vault kv put secret/creds passcode=occlum
Key              Value
---              -----
created_time     2021-08-09T09:41:40.234232551Z
deletion_time    n/a
destroyed        false
version          1
./source_code/bin/Vault kv get secret/creds
====== Metadata ======
Key              Value
---              -----
created_time     2021-08-09T09:41:40.234232551Z
deletion_time    n/a
destroyed        false
version          1
```

```
====== Data ======
Key         Value
---         -----
passcode    occlum
```

6 SCONE: Confidential Cloud-Native Computing

SCONE provides a confidential cloud-native computing solution using Intel SGX technology. The SCONTAIN platform can run NGNIX, Redis, and MongoDB within an Intel SGX enclave. The SCONE solution increases the Zero Trust capability by protecting data and secret IP against attackers, even those with administrative access. The applications do not need to be modified. SCONE supports the most popular programming languages like JavaScript, Python - including PyPy, Java, Rust, Go, C, and C++, and also some transitional programming languages like Fortran. Avoiding source code changes helps ensure that applications are independent of the different trusted execution environments. SCONE supports Kubernetes and Docker deployments. In fact, SCONE provides the SCONE Vault solution, which enables Vault to run within an Intel SGX enclave.

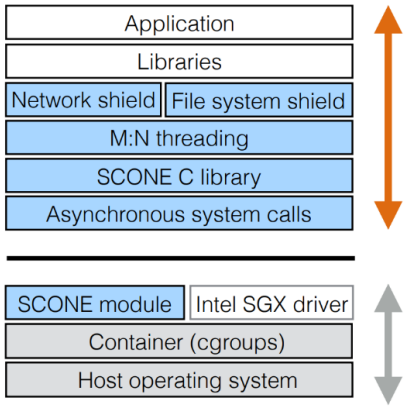


Figure 9. SCONE and Intel SGX in Software Stack

6.1 Run SCONE Vault in a Secure Container

Before running SCONE Vault, install Docker engine and Docker Compose.

Step 1: Clone the demo repository to your local machine.

```
# git clone https://github.com/scontain/scone-vault
# cd scone-vault
```

Step 2: Create `docker-compose.yml` file.

```
# cat docker-compose.yml
version: '3.2'
services:
    vault:
        image: registry.scontain.com:5050/sconecuratedimages/apps:vault-1.5.3-alpine-scone5
        command: sh -c "cd build_dir && ./start_vault.sh"
        environment:
         - VAULT_DEV_ROOT_TOKEN_ID=RootToken
         - SCONE_MODE=hw
        volumes:
         - ./:/build_dir
        cap_add:
         - IPC_LOCK
        devices:
         - "/dev/sgx/enclave"
         - "/dev/sgx/provision"
    scone-vault-nginx:
        image: registry.scontain.com:5050/sconecuratedimages/apps:nginx-1.14.2-alpine-scone5
        environment:
         - URL="http://vault:8200"
         - INDEX=nginx
         - VAULT_ADDR="http://vault:8200"
         - TOKEN=RootToken
         - SCONE_VERSION=1
         - SCONE_MODE=sim
        command: sh -c "cd build_dir && ./install-deps.sh && ./bench.sh"
        volumes:
```

```
      - ./:/build_dir
    depends_on:
      - vault
    devices:
      - "/dev/sgx/enclave"
      - "/dev/sgx/provision"
```

Step 3: Start Vault server and inject some secrets used for NGINX.

```
# docker-compose down
Removing network scone-vault_default
WARNING: Network scone-vault_default not found.
# docker-compose up
Creating network "scone-vault_default" with the default driver
Pulling scone-vault-nginx (registry.scontain.com:5050/sconecuratedimages/apps:nginx-1.14.2-
alpine-scone5)...
nginx-1.14.2-alpine-scone5: Pulling from sconecuratedimages/apps
……
……
scone-vault-nginx_1  |  Test succeeded!, nginx is running with configuration from Vault.
scone-vault-nginx_1  | + CODE=200
scone-vault-nginx_1  | + '[' 200 -ne 200 ]
scone-vault-nginx_1  | + echo -e '\tTest succeeded!, nginx is running with configuration from
Vault.'
scone-vault-nginx_1  | + date '+%s%6N'
scone-vault-nginx_1  | + endNginx=1613143179818731
scone-vault-nginx_1  | + runtime=431795
scone-vault-nginx_1  | + totalTime=910153
scone-vault-nginx_1  | + echo 431795
scone-vault-nginx_1  | + echo 910153
scone-vault_scone-vault-nginx_1 exited with code 0
```

6.2 Run SCONE Vault with Azure Confidential VM

Before running SCONE Vault with Azure confidential VM, you must first create a DCsv2-series VM. See the Azure documentation to create one via the Azure portal. The recommended VM size is DC4s_v2 (4 vCPUs and 16 GB of RAM). Inside the virtual machine, install Docker and `docker-compose`. You do not need to install the Intel SGX driver, as these machines already come with an Intel SGX DCAP driver installed in `/dev/sgx`.

Step 1: Clone the GitHub demo repository.

```
# git clone https://github.com/scontain/scone-vault
# cd scone-vault
```

Step 2: Determine the Intel SGX device for Docker Compose.

```
source sgxdevice.sh && determine_sgx_device
```

Step 3: Start the Vault server and inject some secrets used for NGINX.

```
# docker-compose down
Removing network scone-vault_default
WARNING: Network scone-vault_default not found.
# docker-compose up
Creating network "scone-vault_default" with the default driver
Pulling scone-vault-nginx (registry.scontain.com:5050/sconecuratedimages/apps:nginx-1.14.2-
alpine-scone5)...
nginx-1.14.2-alpine-scone5: Pulling from sconecuratedimages/apps
……
……
scone-vault-nginx_1  |  Test succeeded!, nginx is running with configuration from Vault.
scone-vault-nginx_1  | + CODE=200
scone-vault-nginx_1  | + '[' 200 -ne 200 ]
scone-vault-nginx_1  | + echo -e '\tTest succeeded!, nginx is running with configuration from
Vault.'
scone-vault-nginx_1  | + date '+%s%6N'
scone-vault-nginx_1  | + endNginx=1613143179818731
scone-vault-nginx_1  | + runtime=431795
scone-vault-nginx_1  | + totalTime=910153
scone-vault-nginx_1  | + echo 431795
scone-vault-nginx_1  | + echo 910153
scone-vault_scone-vault-nginx_1 exited with code 0
```

7 Summary

Zero Trust and confidential computing are the new technology trends, with early adoption by enterprises that seek advanced security in solution deployments. Embracing these new technologies can further enhance enterprise security posture and execution. Protection against cyberthreat is critical in this new digital world. This document demonstrates the security capabilities of the 3rd Gen Intel Xeon Scalable processor, such as Intel TME, cryptographic instruction sets, and Intel SGX. These are significant capabilities offered in Intel processors. To get the most out of processor-based security technology, enterprise IT and software solution providers need to choose the right software for the system design. HashiCorp Vault and SCONE confidential cloud-native computing are important software ingredients to secure enterprise applications and data.

Appendix A Platform Configuration

Name	n2-std-8 2nd Gen Intel® Xeon® Scalable Processor	n2-std-8 3rd Gen Intel® Xeon® Scalable Processor
Time	Mon Feb 24 06:27:51 UTC 2022	Mon Feb 24 06:27:51 UTC 2022
Manufacturer	Google	Google
Product Name	Virtual Machine	Virtual Machine
BIOS Version	Google V 1.0	Google V 1.0
OS	Ubuntu 20.04.3 LTS	Ubuntu 20.04.3 LTS
Kernel	5.11.0-1026-gcp	5.11.0-1026-gcp
Microcode	0xffffffff	0xffffffff
IRQ Balance	Disabled	Disabled
CPU Model	Intel® Xeon® CPU @ 2.80 GHz	Intel® Xeon® CPU @ 2.60 GHz
Base Frequency	2.8 GHz	2.6 GHz
CPU Family	6	6
CPU Model	85	106
vCPUs	8	8
Threads per Core	2	2
Cores per Socket	4	4
Sockets	1	1
NUMA Nodes	1	1
Turbo	Disabled	Disabled
Installed	32 GB	32 GB
Automatic NUMA Balancing	Disabled	Disabled

Software Configuration	Software version	Location
Host OS	Ubuntu 20.04.3 LTS	https://ubuntu.com/
Kernel	5.11.0-1026-gcp	https://www.kernel.org/
Vault	Vault 1.9.3	https://www.vaultproject.io/
Docker	Docker 19.03.13	https://www.docker.com/
SCONE	SCONE Vault	https://github.com/scontain/scone-vault
Benchmarking scripts		https://github.com/hashicorp/vault-guides/tree/master/operations/benchmarking/wrk-core-vault-operations

0323/DN/WIT/PDF 691286-002US