Running Linux in a Shielded VM

Michael Kelley
Principal Software Engineer
Microsoft Corporation
Introduction

Linux Shielded VMs

Capability of the Hyper-V hypervisor
Originaly introduced for Windows guests, but now works for Linux guests as well
A mix of Windows/Hyper-V technologies and open source Linux technologies

This session focuses on the Linux aspects

Quick overview of the problem (and what’s out of scope)
End-to-end view of the components that enable Linux Shielded VMs (LSVMs)
Drilldown into 6 technical areas
Wrap-up with current status of Linux Shielded VMs
What is the problem?

Virtual machines introduce new security risks

Compromised or malicious fabric admins can access guest virtual machines

Health of hosts not taken into account before running VMs

Tenant’s VMs are exposed to storage and network attacks

VMs don’t benefit from hardware-rooted security such as TPMs
Shielded VMs: Security Assurance Goals

Encryption of data, both at-rest & in-flight
Virtual TPM enables the use of disk encryption within a VM (e.g. dm-crypt, BitLocker)
Both Live Migration and VM state are encrypted

Fabric admins locked out
Host administrators cannot access guest VM secrets (e.g. can’t see disks, video, etc.)
Host administrators cannot run arbitrary kernel-mode code

Attestation of host health required
VMs can only run on “healthy” hosts designated by the VM owner

NOTE: Shielding is not intended as a defense against DoS attacks
End-to-End Flow

1. Host Guardian Service
2. Template disk
3. Shielding Data File
4. Provisioning Process
5. Boot Linux in shielded mode
6. Additional Scenarios

Guarded Hyper-V hosts
Virtual Machine
Host Guardian Service
Drilldown: Guarded Fabric (1 of 2)

Key Goals
Know and trust the hypervisor software you are running on
Prevent the introduction of malware or guest VM observing tools into the hypervisor
Protect VM secrets as VMs are created and initially booted

Host Guardian Service
Attestation of guarded hosts based on hardware TPM
Release keys to run a VM

Guarded Host running Hyper-V
Boot sequence is validated via physical TPM
Code Integrity policy is enabled – only authorized executables can run
Disabled functionality: debug of VM worker process, guest VM console, most host/guest data exchange
Supplies a “virtual” TPM (vTPM) to the guest VM
Supports Virtualization-Based Security (VBS) for protecting vTPM data and other secrets
Users who request to start specific shielded VMs on the host

**Host Guardian Service components:**

**Attestation Service:**
contains information about the expected configuration of guarded hosts. Authorizes only legitimate guarded hosts to run the shielded VMs.

**Key Protection Service:**
contains keys needed for starting shielded VMs. Ensures that a given key is released only if the host is authorized and is in a Guarded Fabric specified by the VM owner.
Drilldown: Linux Template Disk  (1 of 4)

Key Goals
Base level Linux OS installation from which to create a shielded VM
Should be cloneable as the base for multiple VMs
Should not contain any secrets

Creating a Linux Template Disk
Do a standard Linux installation into the VM (typically on a standalone Hyper-V)
Must be in VHDX format as a “Generation 2” VM so can boot using UEFI Secure Boot
Set up root partition as encrypted using LUKS/dm-crypt with a well-known pass phrase
Install VMM agent for Linux to handled specialization
Drilldown: Linux Template Disk (2 of 4)

Run “Isvmprep” tool in the Linux VM

Makes transformations to help protect the boot path
Updates initramfs to get dm-crypt passphrase from a “file” instead of console

- EFI System partition (unencrypted)
  - Isvmload

- Boot partition (encrypted)
  - grub.cfg
  - Linux kernel
  - Initial ramdisk
  - SHIM
  - grub

- Root partition (encrypted)
  - init program
  - Boot scripts
  - Rest of Linux

Encrypted with well-known passphrase
Drilldown: Linux Template Disk (3 of 4)

Run “Template Disk Creation” wizard in Windows

Creates Volume Signature Catalog – hash of all disk blocks in the boot and root partitions

Puts the VSC in the EFI System Partition, and signs it with a certificate you provide

Puts the Provisioning Agent (PA) into EFI System Partition as the active boot loader

---

**EFI System partition (unencrypted)**
- PA*
- Isvmload
- VSC

**Boot partition (encrypted)**
- grub.cfg
- Linux kernel
- Initial ramdisk
- SHIM
- grub

**Root partition (encrypted)**
- init program
- Boot scripts
- Rest of Linux
Drilldown: Linux Template Disk (4 of 4)

Linux Template Disk Summary

1. Hyper-V virtual disk image (VHD(X)) that can be used to create multiple VMs
2. Standard Linux install with LUKS/dm-crypt encryption on boot and root partitions, using a well-known passphrase
3. Microsoft-signed early boot loader (lsvmload) is installed
4. Creator is asserting that the template disk is “good” when the VSC is created
5. VSC is used later to detect (and reject) any subsequent modifications to the disk
Drilldown: Shielding Data File

Contains per-VM data and secrets
Links together components for secure deployment
Created using Shielding Data File Wizard in Windows

- 'root' password
- timezone
- IP address
- ssh private key
- Other per-VM files
- Guarded Fabric #1
- Guarded Fabric #2
- Guarded Fabric #N
- Cert used to sign VSC
- Owner Key

Template virtual disk
Drilldown: Provisioning (1 of 3)

Key Goals

Create the shielded Linux VM, using a clone of the template disk
Set up VM root & boot partitions with unique dm-crypt encryption keys
Make per-VM settings and secrets available to the VM
Set up everything for normal boot process
Don’t let hypervisor admin have access to the encryption keys or per-VM secrets
Drilldown: Provisioning (2 of 3)

1. Copy template disk and attach to a new shielded VM
2. Hyper-V attests its health to HGS; gets keys to unlock vTPM
3. VM does UEFI Secure Boot to Provisioning Agent
4. PA gets Shielding Data File from Hyper-V host

Sub-process:
- PA validates the disk against the VSC
- PA generates random keys and re-encrypts root and boot partitions
- PA stores new keys in ‘sealedkeys’ and seals to vTPM
- PA gets personalization info from Shielding Data and writes to EFI partition
- PA sets boot path to ‘Isvmload’
- PA deletes itself and reboots
Drilldown: Provisioning (3 of 3)

VM disk layout after provisioning is complete

Ready to do normal Linux boot

EFI System partition (unencrypted)

- Isvmload*
- sealedkeys
- specialization.aes

Encrypted with key sealed in the vTPM

Encrypted with LUKS/dm-crypt masterkey for boot partition

*Active boot loader

Boot partition (encrypted)

- grub.cfg
- Linux kernel
- Initial ramdisk
- SHIM
- grub

Each encrypted with a passphrase in 'sealedkeys'

Root partition (encrypted)

- init program
- Boot scripts
- Rest of Linux

Encrypted with key sealed in the vTPM

Encrypted with LUKS/dm-crypt masterkey for boot partition

Encrypted with key sealed in the vTPM
Drilldown: Normal Boot Path (1 of 3)

Key Goals
Prevent tampering with boot path that could expose secrets
Automate supplying dm-crypt passphrase from the vTPM
Don’t modify the code in the normal Linux boot path (shim, grub, kernel)

<table>
<thead>
<tr>
<th>What is modified</th>
<th>What is not modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>• initramfs updated to get dm-crypt passphrase from a file</td>
<td>• Linux shim</td>
</tr>
<tr>
<td>• lsvmload used as a precursor to the normal Linux boot shim</td>
<td>• grub</td>
</tr>
<tr>
<td>• lsvmload inject disk passphrases as a file into virtualized copy of initramfs</td>
<td>• Linux kernel</td>
</tr>
</tbody>
</table>
I/O to encrypted boot partition is mediated by custom UEFI file I/O protocols
initramfs gets dm-crypt passphrases from injected file

*First boot only
Drilldown: Normal Boot Path (3 of 3)

VM disk layout for ongoing operation

- EFI System partition (unencrypted)
  - Isvmload*
  - sealedkeys
  Encrypted with key sealed in the vTPM

- Boot partition (encrypted)
  - grub.cfg
  - Linux kernel
  - Initial ramdisk
  - SHIM
  - grub

- Root partition (encrypted)
  - init program
  - Boot scripts
  - Rest of Linux

Each encrypted with a passphrase in 'sealedkeys'
Drilldown: Additional Scenarios

**Live migration**
Can migrate a Linux VM to other Hyper-V hosts in same Guarded Fabric Disks and VM memory are encrypted in transit vTPM moves with the VM

**Linux updates**
Kernel and initramfs updates work normally – no special actions needed grubx64.efi and shimx64.efi updates must be manually recopied to new location

**Non-root disks**
Managing the encryption of additional disks is outside the scope of Shielded VMs Use normal dm-crypt techniques
Wrap-Up
Current Status

Hyper-V is updated to enable Linux Shielded VMs
Updated Hyper-V release ships later this fall

Works with Linux distros that can UEFI Secure Boot
We’re collaborating with commercial distro vendors to ensure official support
Red Hat, SUSE, and Canonical

Windows Shielded VMs commercially available today
Rackspace, brightsolid (UK)
Expect to see Linux Shielded VMs commercially available after Hyper-V update ships
Open Source

github.com/Microsoft/lsvmtools

lsvmload – precursor boot loader
lsvmprep – script and tools for creating a Linux template disk
Wrap-Up

Linux Shielded VMs are working end-to-end

Great input from the community over the past year

Made significant changes based on that feedback
Ongoing feedback is always welcome

Questions?