Secure Boot for Windows Virtual Machines on Proxmox

Guide on how to add Secure Boot and Trusted Platform Module (TPM) support for Proxmox Windows 11 Virtual Machines.

September 16, 2021 **In Debian, Linux, Proxmox**

Note: Any and all comments/improvements are welcomed.

What is Secure Boot?

Secure Boot is a security standard that helps make sure that a device boots using trusted software. This feature and the underling hardware Trusted Platform Module (TPM) is also required by Windows to enable certain features such as Bit Locker disk encryption.

Secure Boot and Windows 11

Whist this feature has been around since Microsoft Windows 8, it has gained a lot of coverage in the press because initially Microsoft stated that Microsoft Windows 11 would only install on machines that supported and had Secure Boot enabled. Currently it is unknown if Microsoft will eventually require Secure Boot for Windows 11.

It could be said that using Secure Boot with a Virtual Machines is pointless, however, certain corporate environments require features like Bit Locker to be enabled for a machine to be “compliant” and join their corporate network.

Proxmox and Secure Boot

Proxmox 7.x does not currently support Secure Boot but there has been a significant amount of work done to enable it in the underlying QEMU open source machine emulation and virtualization technology.

This walkthrough leverages that work to provide an updated OVMF UEFI virtual machine bios and a Virtual TPM to support it.

This process is by no means integrated into the Proxmox web interface, virtual machines need to be started from the shell.

Note: This guide is based on the excellent work by the following people, many thanks for all your hard work.

- vTPM and Secureboot capability in a Proxmox-KVM by zimsneexh (https://www.reddit.com/r/Proxmox/comments/oai5cr/guide_vtpm_and_secureboot_capability_in_a/)
- Common EDK II Build Instructions for Linux (https://github.com/tianocore/tianocore.github.io/wiki/Common-instructions)
- Software emulation of a Trusted Platform Module by rayures (https://github.com/rayures/vTPM)
- (https://github.com/stefanberger/)libtpms & swtpm by Stefan Berger (https://github.com/stefanberger/)

The Installation Script

The installation script downloads source from Github, compiles a new version of **OVMF** with to provide the SecureBoot and TPM capability; compiles and installs both **swtpm** and **libtpms** for emulating a TPM2. All of this needs to be run as root.

The script can be run in a couple of different ways – The following command downloads the script and pipes the output directly to the shell.

This is commonly thought of as a security concern and is not recommended without reading and fully understanding the contents of the script.

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/garjones/gareth.com/master/vTPM-install.sh)"
```

If you are uncomfortable with the above, the script can be downloaded from here – https://github.com/garjones/gareth.com/blob/main/vTPM-install.sh (https://github.com/garjones/gareth.com/blob/main/vTPM-install.sh)

Running the Script

The script is as automated as possible and requires very little user input. However during the **checkinstall** phase of swtpm, a number of variables are not enumerated properly and I am yet to work out how to solve this – If anyone has any pointers please leave me a comment.

```
checkinstall 1.6.3, Copyright 2010 Felipe Eduardo Sanchez Diaz Duran
           This software is released under the GNU GPL.
```

```
The package documentation directory ./doc-pak does not exist.
Should I create a default set of package docs?  [y]:
```

Press [Enter] to create the default set of package docs.

```
Preparing package documentation...OK

Please write a description for the package.
End your description with an empty line or EOF.
>>
```

Either enter a description or leave blank and Press [Enter][Enter]

```
*****************************************
**** Debian package creation selected ***
*****************************************

This package will be built according to these values:

0 -  Maintainer: [ root@vtpm ]
1 -  Summary: [ TPM Emulator
Private libraries for swtpm TPM emulators
Include files for the TPM emulator's CUSE interface for usage by clients
Tools for the TPM emulator
Tools for creating a local CA based on a pkcs11 device ]
2 -  Name:    [ swtpm ]
3 -  Version: [ 0.7.0 ]
4 -  Release: [ 1%{?dist} ]
5 -  License: [ GPL ]
6 -  Group:   [ checkinstall ]
7 -  Architecture: [ amd64 ]
8 -  Source location: [ swtpm ]
9 -  Alternate source location: [  ]
10 - Requires: [ trousers >= 0.3.9 bash gnutls-utils
tpm2-pkcs11 tpm2-pkcs11-tools tpm2-tools tpm2-abrmd
expect gnutls-utils trousers >= 0.3.9 ]
11 - Recommends: [  ]
12 - Suggests: [  ]
13 - Provides: [ swtpm ]
14 - Conflicts: [  ]
15 - Replaces: [  ]

Enter a number to change any of them or press ENTER to continue:
```

The Release number contains illegal characters and needs to be cleared:

Note: This is a really bad idea but makes it work.

```
4 [ENTER] [ENTER]
```

Requires contains illegal characters. This is not a good practice, but clearing it solves the problem:

Note: This is a really, really bad idea but makes it work.

```
10 [ENTER] [ENTER]
```

Building can now continue ..

```
[ENTER]
```

Verifying success

On completion if everything has gone correctly the folder /root/vTPM will contain the following files:

- **libtpms_xxx_amd64.deb** – Debian package for libtpms
- **swtpm_xxx_amd64.deb** – Debian package for swtpm
- **OVMF.fd** – Custom OVMF UEFI Bios
- **vTPM-launch.sh** – Custom script to launch a VM with SecureBoot

At this point the only files that are needed are **OVMF.fd** and **vTPM-launch.sh**, everything else can be deleted if you want to.

Starting your VM

The launch script creates a temporary copy of your Virtual machine configuration and modifies it to use the new custom OVMF bios and adds entries to connect to the virtual TPM. It then starts the **swtpm** socket service and finally launches the virtual machine. The socket service runs in the background until the virtual machine is stopped.

First, configure the virtual machine in Proxmox as you would normally making a note of the VMID (101 etc).

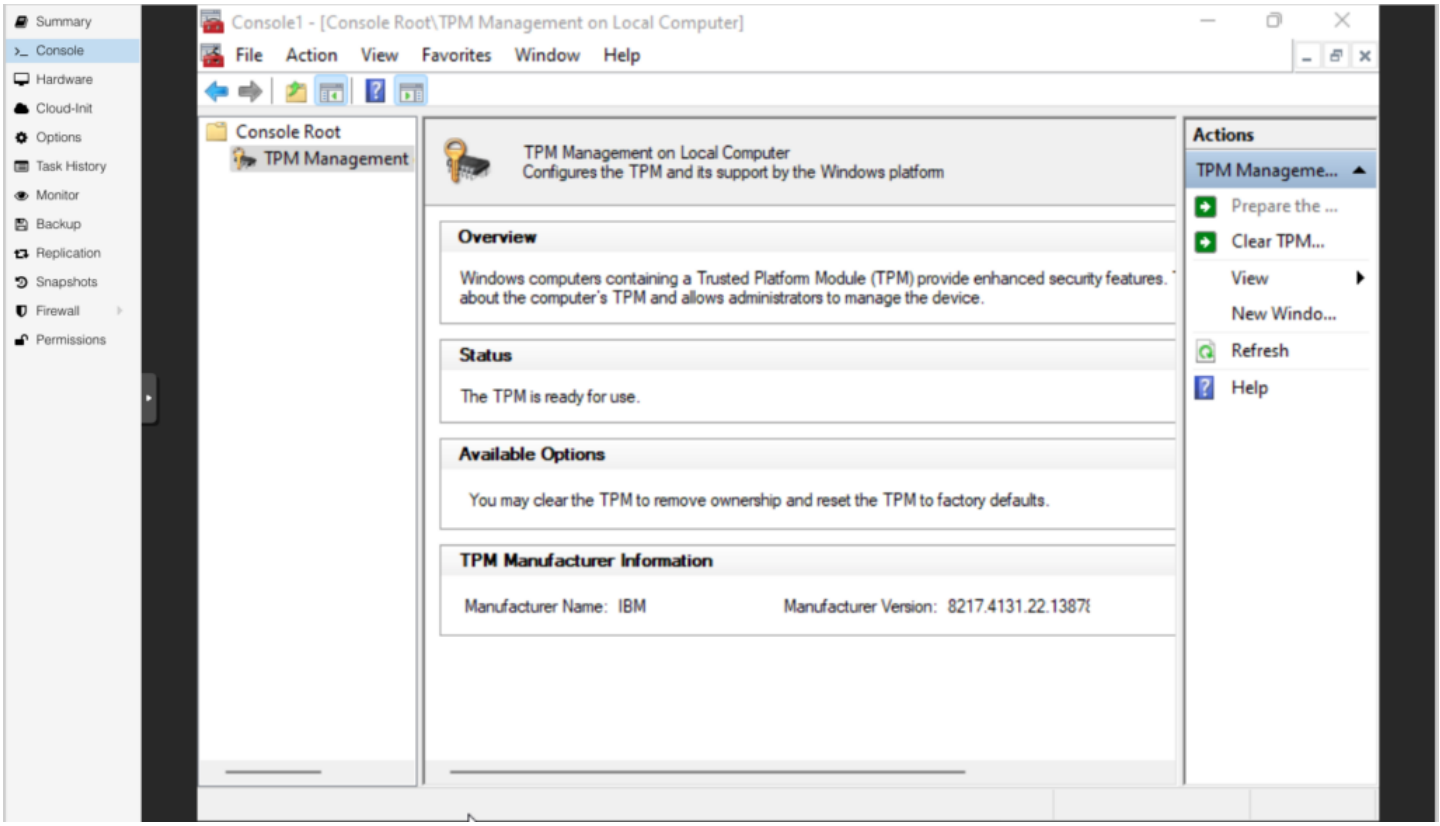
Next run the script passing in the VMID

```
./vTPM-launch.sh 201
```

Looking at the Proxmox console you will see that the machine is using the custom OVMF UEFI bios.

Proxmox vTPM boot screen

After Windows is booted, opening the TPM Microsoft Management Console shows that Windows recognizes a valid TPM!



TPM Management MMC

Job done.

References:
1. [Guide] vTPM and Secureboot capability in a Proxmox-KVM [For Windows 11] – https://www.reddit.com/r/Proxmox/comments/oai5cr/guide_vtpm_and_secureboot_capability_in_a/ (https://www.reddit.com/r/Proxmox/comments/oai5cr/guide_vtpm_and_secureboot_capability_in_a/)
2. Common EDK II Build Instructions for Linux – https://github.com/tianocore/tianocore.github.io/wiki/Common-instructions (https://github.com/tianocore/tianocore.github.io/wiki/Common-instructions)
3. VTPM [rayures] – https://github.com/rayures/vTPM (https://github.com/rayures/vTPM)
4. Stefan Berger – libtpms and swtpm – https://github.com/stefanberger/ (https://github.com/stefanberger/)
4. Gareth Jones – https://github.com/garjones/gareth.com/blob/main/vTPM-install.sh (https://github.com/garjones/gareth.com/blob/main/vTPM-install.sh)

Leave a Reply

Comment

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

Post Comment