Guidance Supplement

VMware ESXi 6.7 Update 2 with 6.7 Patch Version 201905001

Common Criteria (CC) Evaluation with Protection Profile (PP) for Virtualization Version 1.0 with Server Virtualization Extended Package (EP) 1.0

Document Version: 1.15
Document Date: November 7, 2019
## Revision History

<table>
<thead>
<tr>
<th>Ver #</th>
<th>Description of changes</th>
<th>Modified by</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Initial release of document</td>
<td>Kevin Christopher</td>
<td>April 22, 2019</td>
</tr>
<tr>
<td>1.01</td>
<td>Updated copyright date</td>
<td>Kevin Christopher</td>
<td>April 30, 2019</td>
</tr>
<tr>
<td>1.02</td>
<td>Updated final name of Patch Release</td>
<td>Kevin Christopher</td>
<td>May 17, 2019</td>
</tr>
<tr>
<td>1.03</td>
<td>Added CAVP and CMVP certificates for vmkernel module loaders</td>
<td>Kevin Christopher</td>
<td>June 17, 2019</td>
</tr>
<tr>
<td></td>
<td>Added SSH (server) configuration instructions (section 2),</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>updated configuration descriptions (section 4).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Listed TLS ciphers explicitly.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.04</td>
<td>4.2.8: Default SSH configuration is not compatible with protection profile; corrected name to “Extended Package for SSH”</td>
<td>Kevin Christopher</td>
<td>June 19, 2019</td>
</tr>
<tr>
<td></td>
<td>4.2.9: syslog TLS client is not configurable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.05</td>
<td>Updated AGD title to match ST title</td>
<td>Kevin Christopher</td>
<td>July 18, 2019</td>
</tr>
<tr>
<td></td>
<td>4.1: Updated remote syslog description</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.3.4: Indicated AMD is not covered by NIAP evaluation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.3.1: Added link to Host Client for vNIC configuration</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.1: updated syslog RFCs.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.4.2: documented default password complexity</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.5.4 and 4.2.8.3: removed ecdsa-sha2-nistp521</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.5.3, 4.6.1: added Host Client instructions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.06</td>
<td>Updated 2.6 and 2.7 covering operating modes and support</td>
<td>Kevin Christopher</td>
<td>July 19, 2019</td>
</tr>
<tr>
<td></td>
<td>Expanded 1.3 assumptions</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.3.2.1 and 4.5.3: added Host Client instructions which are absent from public documentation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.07</td>
<td>4.4.1: additional detail on handling of login failures</td>
<td>Kevin Christopher</td>
<td>August 6, 2019</td>
</tr>
<tr>
<td>1.08</td>
<td>4.4.1: corrections on handling of login failures</td>
<td>Kevin Christopher</td>
<td>August 12, 2019</td>
</tr>
<tr>
<td>1.09</td>
<td>4.4.1: minor wording adjustment</td>
<td>Kevin Christopher</td>
<td>August 20, 2019</td>
</tr>
<tr>
<td></td>
<td>4.3.2.3: added section on physical CD-ROM/DVD</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.4: removed exclusion of local CD-ROM from evaluation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.10</td>
<td>2.4.5: security mitigations which administrators may need to apply</td>
<td>Kevin Christopher</td>
<td>September 5, 2019</td>
</tr>
<tr>
<td></td>
<td>2.8: known security issues since last ESXi update release (ESXi 6.7 Update 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.6.1: Host Client cannot install patches, included commands to display VIB versions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.11</td>
<td>Editorial corrections</td>
<td>Kevin Christopher</td>
<td>September 6, 2019</td>
</tr>
<tr>
<td>1.12</td>
<td>Added VMSAs from September</td>
<td>Kevin Christopher</td>
<td>October 11, 2019</td>
</tr>
<tr>
<td></td>
<td>5.2.2.3: added section addressing key destruction in vmkernel</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.4.1: added information on how to authenticate host</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ver #</td>
<td>Description of changes</td>
<td>Modified by</td>
<td>Date</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------------------------------------------------------------</td>
<td>------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>1.13</td>
<td>Removed several CMVP mentions, due to CMVP delay</td>
<td>Kevin Christopher</td>
<td>October 15, 2019</td>
</tr>
<tr>
<td>1.14</td>
<td>5.2.1: CMVP certificate number now available 2.8.1.8: VMSA-2019-0019 discussion 4.4.1 and 5.4.2: Additional password information 5.2.7: ESXi does not implement TLS mutual authentication</td>
<td>Kevin Christopher</td>
<td>November 5, 2019</td>
</tr>
<tr>
<td>1.15</td>
<td>4.4.1: Clarify CGI login double-count, added warning 5.2.1: remove all CMVP information</td>
<td>Kevin Christopher</td>
<td>November 7, 2019</td>
</tr>
<tr>
<td>1.16</td>
<td>5.2.1: remove CMVP reference, added CAVP reference</td>
<td>Ravi Jagannathan</td>
<td>November 8, 2019</td>
</tr>
</tbody>
</table>
# Table of Contents

1 Introduction .................................................................................. 8  
1.1 Purpose .................................................................................... 8  
1.2 Document Reference ................................................................. 8  
1.3 Assumptions ............................................................................. 9  
1.4 Features and Functions Not Included in the TOE Evaluation .......... 9  

2 Installation Guidelines and Preparative Procedures .................... 12  
2.1 Evaluated Configuration ............................................................ 12  
2.2 TOE Components ....................................................................... 13  
2.3 Supporting Environmental Components ...................................... 13  
2.4 Installation of the TOE ............................................................... 13  
2.4.1 Configure Firmware ............................................................. 13  
2.4.2 Obtaining Software ............................................................. 14  
2.4.3 Verify Software .................................................................... 15  
2.4.4 Updating Software ............................................................... 15  
2.4.5 Additional ESXi Configurations ............................................ 15  
2.4.6 Additional device configuration ............................................ 16  
2.5 Configuring the TOE Environmental Components .................... 16  
2.5.1 Installing ESXCLI for Remote Management ............................ 16  
2.5.2 Audit Configuration ............................................................ 17  
2.6 Operating Modes ....................................................................... 18  
2.7 Obtaining Support ..................................................................... 18  
2.8 Recent and Open Security Issues and Mitigations ....................... 18  
2.8.1 VMware Service Advisories (VMSA) ..................................... 18  
2.8.2 OpenSSH ................................................................. 20  

3 Operational Guidance: Advanced Options ................................. 22  
3.1 Configuring Advanced Options Using the UI ............................... 22  
3.2 Configuring Advanced Options Using the VIM API ..................... 22  
3.2.1 Querying Advanced Options ............................................... 22  
3.2.2 Setting Advanced Options .................................................. 22  
3.3 Selected Advanced Options ...................................................... 23  

4 Operational Procedures for Administrators ................................ 25  
4.1 Audit Configuration (FAU) ......................................................... 25  
4.1.1 Viewing Audit Records ....................................................... 25  
4.1.2 Configuring Local Audit Records ....................................... 25  
4.1.3 Configuring Remote Audit Server ...................................... 25  
4.2 Configuration (FCS) ................................................................. 26  
4.2.1 Cryptographic Key Generation ........................................... 26  
4.2.2 Cryptographic Key Establishment ....................................... 28  
4.2.3 Cryptographic Key Destruction ......................................... 28  
4.2.4 Cryptographic Operation ................................................... 28  
4.2.5 Random Bit Generation ..................................................... 29  
4.2.6 Entropy for Virtual Machines ............................................ 29  
4.2.7 HTTPS ............................................................................ 29  
4.2.8 SSH .............................................................................. 29  
4.2.9 TLS Protocol ................................................................. 31  
4.3 Protection of User (VM) Data (FDP) ........................................ 33  
4.3.1 Virtual Networking ............................................................ 33  
4.3.2 Physical Platform Resources ............................................. 35  
4.3.3 Memory Zeroization .......................................................... 38  
4.3.4 Hardware-based VM Isolation .......................................... 38  
4.4 Authentication Configuration (FIA) .......................................... 38
4.4.1 Authentication Failure Handling ..............................................................38
4.4.2 Password Management ........................................................................39
4.4.3 Additional Authentication Mechanisms ..................................................40
4.4.4 X.509 Certificate Validation and Authentication ......................................40
4.5 Management Functions (FMT) ..................................................................41
4.5.1 Default Sharing Configurations .............................................................41
4.5.2 Timeouts ...............................................................................................41
4.5.3 Management APIs (Consolidated) ..........................................................41
4.6 Hypervisor Integrity (FPT) .........................................................................44
4.6.1 Trusted Updates ....................................................................................44
4.6.2 Hypercall Controls ...............................................................................45
4.6.3 Removable Devices and Media ..............................................................45
4.7 Accessing the Hypervisor (FTA) ...............................................................47
4.7.1 Administrative Access Banner .............................................................47
4.8 Secure Communication with the Hypervisor (FTP) ......................................47
4.8.1 Isolating VM Networks from the Management Network .........................47
4.8.2 Configuring Remote Audit Servers .......................................................47
5 Documentation Clarifications ......................................................................48
5.1 Audit Configuration ...................................................................................48
5.1.1 Audit Data and Auditable Events ..........................................................48
5.2 Cryptographic Configuration .....................................................................50
5.2.1 FIPS Certificates ..................................................................................50
5.2.2 Cryptographic Key Destruction ............................................................51
5.2.3 Entropy for Virtual Machines ...............................................................51
5.2.4 HTTPS protocol ....................................................................................52
5.2.5 SSH Protocol (Server) ..........................................................................52
5.2.6 SSH Protocol (Client) ..........................................................................52
5.2.7 TLS Protocol .........................................................................................53
5.3 User Data Configuration ...........................................................................53
5.3.1 Memory Zeroization ............................................................................53
5.3.2 Hardware-based VM Isolation ...............................................................54
5.3.3 Physical Resource Controls .................................................................54
5.4 Authentication Configuration ....................................................................56
5.4.1 Login Mechanisms ...............................................................................56
5.4.2 Password Complexity Enforcement ......................................................56
5.5 Management Functions ...........................................................................57
5.5.1 The CGI Interface ...............................................................................57
5.6 Hypervisor Integrity ..................................................................................58
5.6.1 VMM Isolation .....................................................................................58
5.6.2 Hypercalls ............................................................................................58
5.6.3 Virtual Device Parameters .................................................................58
5.6.4 Virtualization Hardware Assists ............................................................59
5.6.5 Execution Environment Mitigations .......................................................63
5.7 Accessing the Hypervisor .........................................................................65
5.7.1 Administrative Access Banner .............................................................65
5.8 Secure Communication with the Hypervisor .............................................65
5.8.1 The Management Network ..................................................................65
5.8.2 Secure Communication Using the Management Network .....................66
5.8.3 User Interface Indicators ......................................................................67
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Supporting Environmental Components</td>
<td>13</td>
</tr>
<tr>
<td>Table 2</td>
<td>Advanced Options</td>
<td>24</td>
</tr>
<tr>
<td>Table 3</td>
<td>TLS Settings in Configuration Files</td>
<td>33</td>
</tr>
<tr>
<td>Table 4</td>
<td>BackingInfo Types for Virtual Network Devices</td>
<td>35</td>
</tr>
<tr>
<td>Table 5</td>
<td>BackingInfo Types for Virtual USB Devices</td>
<td>36</td>
</tr>
<tr>
<td>Table 6</td>
<td>VIM APIs for Certificate Management</td>
<td>40</td>
</tr>
<tr>
<td>Table 7</td>
<td>Assorted Management Functions</td>
<td>44</td>
</tr>
<tr>
<td>Table 8</td>
<td>BackingInfo Types for Virtual Floppy Devices</td>
<td>46</td>
</tr>
<tr>
<td>Table 9</td>
<td>BackingInfo Types for Virtual CD-ROM Devices</td>
<td>46</td>
</tr>
<tr>
<td>Table 10</td>
<td>Audit Record Parameter Names</td>
<td>50</td>
</tr>
<tr>
<td>Table 11</td>
<td>FIPS Certificates</td>
<td>50</td>
</tr>
<tr>
<td>Table 12</td>
<td>CGI Interfaces</td>
<td>57</td>
</tr>
<tr>
<td>Table 13</td>
<td>PCI IDs</td>
<td>59</td>
</tr>
<tr>
<td>Table 14</td>
<td>IO Device Specification</td>
<td>59</td>
</tr>
<tr>
<td>Table 15</td>
<td>CPU Feature Support in TOE's CPU</td>
<td>63</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1: The TOE Evaluated Configuration........................................................................................................12
1 INTRODUCTION

1.1 Purpose

This document describes the operational guidance and preparative procedures for VMware ESXi™ 6.7 Patch Release ESXi670-201905001, which is intended to install on top of VMware ESXi™ 6.7 Update 2. This document defines the necessary steps to configure the Target of Evaluation (TOE) for use and provides guidance on the ongoing secure usage of the TOE.

1.2 Document Reference

This document serves as a supplement to the standard VMware documentation set, and as such references (either implicitly or explicitly) the following documents:

- VMware vSphere Documentation portal
- VMware ESXi Installation and Setup
- VMware ESXi Upgrade
- VMware vSphere Security
- vSphere Single Host Management – VMware Host Client
- vSphere Virtual Machine Administration
Whitepapers describing ESXi architecture:

- Security of the VMware vSphere Hypervisor

- Timekeeping in VMware Virtual Machines

This document also references the following API and CLI command set documentation:

- vSphere Management SDK, version 6.7
  - [https://code.vmware.com/web/sdk/6.7/vsphere-management](https://code.vmware.com/web/sdk/6.7/vsphere-management)

- vSphere Web Services API Reference (describes the VIM API)
  - [https://code.vmware.com/apis/358/vsphere](https://code.vmware.com/apis/358/vsphere)

- vSphere Command Line Reference (describes the ESXCLI commands)

1.3 Assumptions

The following assumptions are made with regards to the setup, installation, and ongoing operation.

- The platform has not been compromised prior to installation of ESXi.
- The ESXi system is provided with appropriate physical security.
- The IT environment specified in the Security Target (ST) is assumed to be properly implemented by a trained and competent administrator.
  - The administrator uses only designated interfaces to manage ESXi.
  - The administrator regularly installs software updates per VMware guidance.
  - The administrator follows guidance in Section 2 to properly configure ESXi security policies during to deployment.
  - The administrator follows guidance throughout this document to properly maintain ESXi security policies during normal operation.
- The administrator deploys workloads (virtual machines) appropriate to the risks of covert channels inherent in shared resources. For example, two virtual machines deployed to a single datastore carry a covert channel risk that information can be exchanged through high/low bandwidth usage on the datastore.
- The IT environment prevents willfully negligent or hostile actions from an administrator.

1.4 Features and Functions Not Included in the TOE Evaluation

Features that are not part of the evaluated configuration of the TOE are:

- AMD CPUs or Intel CPUs other than the Intel Xeon 6126. Support for other CPUs is implemented by ESXi but this evaluation covers only the Intel Xeon 6126.
- Trusted Platform Module (TPM) devices on the host. Usage is implemented but not evaluated.
• Virtual TPM devices attached to virtual machines. These devices are believed to be correct, however, the cryptographic implementation is not FIPS-validated.
• Secure boot functionality. Though secure boot is required in the NIAP-validated configuration for its side effects, the implementation makes use of cryptography, which is not FIPS-validated and thus no claim is made about the additional security of secure boot.
• VIM Roles and Permissions. For the purposes of this evaluation, all VIM users are “Administrator” with full VIM permissions, and no lower-privileged roles are evaluated. This is to reduce the scope of evaluation in a way that reflects typical usage by vCenter Server, which always operates with full permissions.

Functionality that is disallowed in the TOE Evaluation:
• Active Directory integration for user account management. The NIAP-validated configuration uses only a local database of user accounts.
• RDM passthrough of storage LUNs. The isolation of raw device mappings is not covered by the evaluation as RDMs are not present in the TOE configuration.
• PCI Passthrough (DirectPath I/O). Testing PCI passthrough configurations is not covered by the evaluation as no devices suitable for PCI passthrough are present in the TOE configuration.
• vGPU passthrough of graphics cards. The passthrough mechanism used for vGPU does not provide isolation between virtual machines.
• Virtual Shared Disks (Multiwriter disks). Shared virtual disks are not covered by the evaluation and requires explicit Administrator action to enable.

Functionality that can be used but is not evaluated in the TOE:
• vCenter Server. vCenter Server manages an ESXi host programmatically, using the VIM API evaluated here. Thus, managing ESXi through vCenter Server is not incompatible with this TOE Evaluation but is also not evaluated.
• vSAN and NSX. vSAN is included but not enabled in the TOE due to being distributed as a separate license. NSX is installed as a separate product and is not included in the TOE. Usage of either product is not covered by this evaluation.

vSphere offers several well-known features often desired for operational or security reasons. The following discussion describes features which are disabled in the TOE, not evaluated for NIAP, and are not included in a NIAP-validated environment.
• **vMotion** (including **SvMotion** and **XvMotion**)  
  Not evaluated by NIAP, disallowed in a NIAP-validated environment by closing firewall ports. The vMotion wire protocol itself is not a trusted path and needs external mechanisms, possibly including physical network isolation, to establish a trusted path.
• **Virtual Machine Encryption**  
  Virtual Machine Encryption is included in the TOE, with the exception of the implementation of the virtual TPM which is not covered by FIPS cryptography. The security of virtual machine encryption is not evaluated due to lack of NIAP Virtualization Protection Profile assurance activities requiring encryption of virtual machine state.
• **IPSec**  
  IPSec is not included in the TOE and is not evaluated for NIAP. IPSec is disabled by default and requires explicit configuration to enable, which should not be performed in a NIAP-validated environment. This exclusion reflects infrequent usage of IPSec in practice.
• **CIM** (Common Information Model) and **SNMP** (Simple Network Management Protocol)
CIM and SNMP are not included in the TOE and are not evaluated for NIAP. These services are disabled by default and require explicit configuration to enable, which should not be performed in a NIAP-validated environment. This exclusion reflects limitations of the software packages used to implement CIM. The SLP service is enabled by default; SLP is read-only and not sensitive, and section 2.4.5 covers disabling SLP.

- **Active Directory** integration for user account management.
  Active Directory integration is not included in the TOE and not evaluated for NIAP. Active Directory integration must be left unconfigured in a NIAP-validated environment. This exclusion reflects usage of non-FIPS-validated cryptography.
2 INSTALLATION GUIDELINES AND PREPARATIVE PROCEDURES

2.1 Evaluated Configuration

The evaluated configuration consists of a single instance of VMware ESXi and associated environmental components.

ESXi TOE

![Diagram of ESXi TOE]

Figure 1: The TOE Evaluated Configuration

The evaluated configuration allows and authenticates incoming connections through the following interfaces:

- **VIM API**: Remote procedure call using SOAP (XML) over HTTPS. Authentication occurs using specific SOAP RPCs, for example `SessionManager.Login`, including HTTP cookies set by that interface.
- **ESXCLI**: Invocation of specific commands using a CGI interface (HTTPS POST + reply). The command and parameters are encoded into the HTTP request. Authentication occurs using encoded parameters representing username and password.
- **CGI**: Invocation of specific commands exposed as URLs (HTTPS GET, PUT, and POST). Authentication occurs using HTTP Basic Auth (over HTTPS) or an HTTP session cookie (over HTTPS).
- **SSH**: Login using the SSH protocol.
- **DCUI**: Physical access to the host platform using keyboard and screen. Authentication occurs with username and password.

The evaluated configuration establishes outgoing connections through the following interfaces:

2.2 TOE Components

The TOE consists of the ESXi hypervisor. For the TOE, the hypervisor is installed on top of a physical system (Dell PowerEdge R740) composed of CPUs (Intel Xeon 6126 “Skylake”), local storage and network adapters, and a keyboard and a monitor.

The ESXi hypervisor is made up of a kernel (“VMkernel”), which runs virtual machines (“VM”), and a management agent (“Host Agent”). The Host Agent implements the VIM API, ESXCLI, and a few assorted CGI commands (all HTTPS) over the network. The hypervisor also provides an SSH Server implementation (“SSHD”), which implements the SSH protocol over the network. The hypervisor can communicate with a remote syslog server over the network. The hypervisor includes a very limited local (physical console) user interface (“DCUI”), which primarily exists to configure networking and restart the host. The DCUI does not provide any access to virtual machines.

The hypervisor includes the Host Client, a web-based user interface implemented as an HTML5-based single-page application (JavaScript running in a web browser). The Host Client is implemented exclusively using calls to the VIM API and various CGI endpoints documented in the following sections. As a result, within the TOE, evaluation is performed on the VIM API with the assumption that anything the Host Client does is performed using that evaluated VIM API.

2.3 Supporting Environmental Components

The ESXi hypervisor is designed for self-contained operation. Few external components are needed.

<table>
<thead>
<tr>
<th>Components</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux system for management</td>
<td>System from which to make VIM API calls or ESXCLI calls to configure and manage ESXi. Any Linux distribution released after around 2014 should be sufficient. RHEL7 and Ubuntu 16.04 have been specifically tested. NOTE: In many deployments, vCenter Server is used to make VIM API calls.</td>
</tr>
<tr>
<td>Remote Syslog Server</td>
<td>A server which implements RFC 3164 “The BSD Syslog Protocol” and RFC 5425 “TLS Transport Mapping for Syslog.” NOTE: A remote syslog server is optional. However, usage of a remote syslog server is included in the NIAP-validated configuration.</td>
</tr>
</tbody>
</table>

Table 1: Supporting Environmental Components

2.4 Installation of the TOE

General installation instructions are available for ESXi at the following location. https://docs.vmware.com/en/VMware-vSphere/6.7/com.vmware.esxi.install.doc/GUID-B2F01BF5-078A-4C7E-B505-5DFFED0B8C38.html

2.4.1 Configure Firmware

Before installing the ESXi software, configure the system’s firmware to support a secure hypervisor.
On the TOE system (a Dell PowerEdge R740), press F2 during boot to enter System Setup. The following settings must be configured:

1. Ensure Boot Mode is set to UEFI (instead of Compatibility Mode BIOS). The firmware will require this be set for Secure Boot. Many systems default to this setting and might require a reboot after the setting is changed.
2. Enable Secure Boot. (System Security / SECURE BOOT, leave Secure Boot Policy as Standard.)
3. Confirm the following settings, which are generally enabled by default:
   a. VT enabled (Hardware Virtualization)
   b. VT-d enabled (IOMMU)
4. Boot Order. During installation, the system must boot from the optical drive with installation media. After installation, this can be changed to boot from the physical disk.

**NOTE:** Secure boot is itself not validated by the NIAP process due to usage of non-FIPS-validated cryptography during boot. Instead, using Secure Boot forces additional security checks when installing software updates (for example, unconditionally validating code signatures) that are necessary to satisfy NIAP requirements.

### 2.4.2 Obtaining Software

You can obtain software either from an OEM or from the “My VMware” download portal at:

[https://my.vmware.com/](https://my.vmware.com/)

For an evaluation copy of ESXi, go to:


For product patches to ESXi, see [KB 1021623](https://kb.vmware.com/kb/1021623) or go to:

[https://my.vmware.com/group/vmware/patch](https://my.vmware.com/group/vmware/patch)

The software selected must be at least ESXi 6.7 Patch Release ESXi670-201905001 to contain all functionality necessary for a NIAP-evaluated configuration. Obtain this version by installing a previous version of ESXi 6.7 (installation of ESXi 6.7 Update 2 is recommended) and then applying the ESXi 6.7 Patch Release. Direct installation of a Patch Release is not supported. ESXi 6.7 Patch Release ESXi670-201905001 is based on the ESXi 6.7 Update 2 release. VMware does not support a NIAP-validated configuration on any software version prior to ESXi 6.7 Update 2 with the specified Patch Release installed. Later versions of ESXi 6.7 (subsequent Patch Releases, updates subsequent to Update 2) will maintain this NIAP-validated configuration.

You can install the ESXi 6.7 Update 2 software in the following ways:

- **Physical media:** Using another system, burn the ESXi 6.7 ISO to a CD-ROM.
- **Virtual media:** Some systems have an iDRAC or iLO interface (on a separate IP address), which can be used as a “remote console” to manage the host. The iDRAC or iLO can be configured to present the ESXi 6.7 ISO through firmware.
- **USB:** Described in *VMware ESXi Installation and Setup* documentation, not covered here.
- **PXE:** Described in *VMware ESXi Installation and Setup* documentation, not covered here.

**NOTE:** PXE booting requires transferring sensitive system state using FTP or HTTP during boot and is not recommended for a secure installation, unless external measures are taken to ensure the security of the network used for PXE booting. Such measures are beyond the scope of this document.
To install an ESXi 6.7 Patch Release, follow guidance in section 4.6.1. Briefly:

1. Be running ESXi 6.7 of some previous version.
2. Put the host in maintenance mode.
3. Use `esxcli software vib install` or `esxcli software vib update` to install the patch. The `install` command will forcibly install all contents of the specified patch; the `update` command will install only newer content.
4. Exit maintenance mode and reboot the system.
5. Confirm installation of the patch by verifying build number. See section 2.4.3.

Note: patches of the core ESXi software always require a reboot.

### 2.4.3 Verify Software

To verify the installed ESXi software version, as an administrator, perform the following:

- Using SSH, run the command `vmware -v` to determine the build number.

For more information about determining the installed software version, see the VMware knowledge base article at [https://kb.vmware.com/s/article/1022196](https://kb.vmware.com/s/article/1022196).

### 2.4.4 Updating Software

If the TOE is not running the evaluated version of software, contact VMware, Inc. to obtain the appropriate license grant for the evaluated software version.

See section 4.6.1 for ESXi update instructions.

### 2.4.5 Additional ESXi Configurations

ESXi requires a few non-default parameters to be set for a NIAP configuration. Though ESXi attempts to be secure by default, some of these options come at a performance cost or disable common features and thus are not enabled by default.

To adjust SSH (server) configuration (see section 4.2.8), modify the file `/etc/ssh/sshd_config` in the following ways:

- Modify the `Ciphers` line to remove `aes192-ctr`
- Add the following lines to configure allowed key exchanges and public key algorithms:
  ```
  KexAlgorithms diffie-hellman-group14-sha1,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521
  PubKeyAcceptedKeyTypes ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,rsa-sha2-256,rsa-sha2-512,ssh-rsa
  ```

To enable eager memory zeroing (to minimize lifetime of cryptographic keys in memory, see section 4.2.3.1):

```bash
esxcli system settings advanced set -o "/Mem/MemEagerZero" --int-value "1"
```

To disable features and associated services, close ports using firewall configuration:

```bash
esxcli network firewall ruleset set -e false -r "CIMSLP"
esxcli network firewall ruleset set -e false -r "DVSSync"
esxcli network firewall ruleset set -e false -r "faultTolerance"
esxcli network firewall ruleset set -e false -r "iofiltervp"
esxcli network firewall ruleset set -e false -r "vMotion"
```

Note: this configuration will disable Distributed Virtual Switches, Fault Tolerance, SLP, vMotion, and Virtual Machine Encryption. Customers should evaluate their security exposure...
and functionality needs when choosing to disable features. See section 1.4 for further information about why specific features are disabled.

Administrators should also examine section 2.8 for unmitigated security vulnerabilities and associated exposure. Four scenarios in particular require an administrator to choose between security and performance / features.

1) VMSA-2018-0025: 3D Graphics denial-of-service, workaround is to disable 3D graphics in virtual machines. This is the default setting. Depending on 3D graphics requirements of virtual machines, exposure to this denial-of-service may be required.

2) VMSA-2019-0008: Spectre-class issue, workaround is to enable an alternative scheduler that provides isolation at some performance cost. The default setting does not provide side-channel isolation. Depending on performance needs, exposure to side-channel attacks may be required.

3) VMSA-2019-0011: Hostd denial-of-service, workaround using a configuration change, or apply a more recent ESXi patch.

4) VMSA-2019-0019: 3D Graphics denial-of-service, workaround is to disable 3D graphics in virtual machines. This is the default setting. Depending on 3D graphics requirements of virtual machines, exposure to this denial-of-service may be required.

5) OpenSSH: several client-side scp issues, workaround is to disable outbound SSH access. Depending on usage of shell access and trust of SSH accessible servers, mitigation may be unnecessary.

2.4.6 Additional device configuration

Several ESXi features are not evaluated within the TOE due to lack of available devices. The NIAP evaluation expects these features to be disabled, and not merely unused. To configure an ESXi system to ensure these features are unavailable:

- PCI Passthrough: ensure all PCI devices are supported by VMware ESXi and have a suitable VMware-supplied driver loaded so that all PCI devices will be associated with the host only. No devices should be available for association with a virtual machine (see section 4.3.2.4).
- vGPU: ensure nVidia GRID PCI devices are not installed in the TOE.
- RDM passthrough of storage LUNs: ensure all local disks are formatted with VMFS volumes and mounted as datastores. No unmounted storage volumes should be available to the host.

2.5 Configuring the TOE Environmental Components

This section describes the steps involved when configuring the TOE environmental components.

2.5.1 Installing ESXCLI for Remote Management

The ESXCLI command set is available as:

- Part of the vSphere CLI package, which includes CLIs for ESXi and vCenter Server. For more information about the vSphere CLI, see https://code.vmware.com/tool/vsphere-cli/6.7.
Note: There are known issues installing the vSphere CLI package on newer Linux distributions.

- A standalone ESXCLI download.
  https://code.vmware.com/web/tool/6.7u2/esxcli

This guide covers using the ESXCLI standalone download. To install, decompress the provided .tgz file into a directory of your choice (this guide uses /opt/vmware) then add the installed esxcli binary to your path (this guide describes using a symlink). For example:

```bash
su    # Install as root for all users
mkdir /opt/vmware
`tar zxf esxcli-6.7.0-13004787-lin64.tgz -C /opt/vmware
ln -s /opt/vmware/esxcli/esxcli /usr/bin/esxcli
esxcli --version    # Confirm successful installation
```

When installing ESXCLI, note the following:

- Installing as root is not required.
- You can install ESXCLI in a per-user location.
- You can install both vSphere CLI and standalone ESXCLI on the same system, but it is not recommended.

### 2.5.2 Audit Configuration

The following instructions assume ESXCLI is installed on a Linux system suitable for remote management of ESXi.

To enable local auditing, execute the following commands:

```bash
esxcli --server <host> system auditrecords local set --size=4096
esxcli --server <host> system auditrecords local enable
```

To enable remote auditing, execute the following commands (using appropriate values for a CA and host name for the syslog server):

```bash
esxcli --server <host> system security certificatestore add -f cacert.pem
esxcli --server <host> system syslog \ 
    config set --loghost="ssl://syslog.example.com:1514"
```

```bash
esxcli --server <host> system syslog config set --crl-check="true"
```

```bash
esxcli --server <host> system syslog config set --x509-strict="true"
```

```bash
esxcli --server <host> system syslog reload
esxcli --server <host> system auditrecords remote enable
```

See section 4.1 for further details.

CA roots can also be configured using ESXCLI commands:

```bash
esxcli system security certificatestore add --file=<local-file>
esxcli system security certificatestore list
esxcli system security certificatestore remove --issuer=<issuer> --serial=<serial>
```

For more information about the esxcli system command, see the following topic:

2.6 Operating Modes

The ESXi hypervisor functions in several operating modes.

- **Installation.** During installation, security properties have not yet been configured and ESXi may be unable to offer management functionality until fully configured. For example, TLS keys may not yet be provisioned, and thus remote access may be inaccessible. Once installation completes, ESXi is fully usable.

- **Maintenance Mode.** ESXi can enter maintenance mode using the VIM API, the ESXi Host Client, or the DCUI local console. Entering maintenance mode requires that no virtual machines currently be running. While in maintenance mode, virtual machines may not be powered on and attempting to power on a virtual machine will return an error. The intent of this mode is to assist administrators in disallowing conflicting operations while conducting maintenance activities, including applying system updates. While in maintenance mode, ESXi maintains all security properties (no security properties are weakened while in maintenance mode).

- **Normal operation.** All security and management functions are available as described in this document.

2.7 Obtaining Support

In the event of software failure, customers should engage with VMware Global Support Services to make use of any purchased support contract(s).

https://www.vmware.com/support/contacts.html

VMware also offers self-service documentation and knowledge base articles.

https://www.vmware.com/support/vsphere.html

In the event of hardware failure, customers should communicate with the vendor who supplied the hardware platform for support options.

2.8 Recent and Open Security Issues and Mitigations

The ESXi 6.7 Patch Release ESXi670-201905001 is part of an ongoing lifecycle, where security issues are found, resolved, or mitigated. This section discusses several such issues which were either fixed in the evaluated patch release or require mitigation in the current patch release.

2.8.1 VMware Service Advisories (VMSA)

This section describes VMware Service Advisories (VMSAs) applying to ESXi issued following the release of ESXi 6.7 Update 2, or for prior releases with no resolution.

2.8.1.1 VMSA-2018-0025

CVE-2018-6977
This advisory describes that a malicious 3D shader from a guest can put graphics hardware into an infinite loop, causing a denial of service to other virtual machines. This is a consequence of graphics hardware implementation which cannot be mitigated by the hypervisor; VMware expects any resolution to come from updated graphics hardware or graphics vendor drivers. Removing 3D-acceleration from the virtual machine configuration will prevent the issue. 3D acceleration is not enabled for virtual machines on ESXi by default.

Administrators wishing to evaluate risk exposure should consider that WebGL allows a malicious 3D shader to force a web browser to cause the same denial of service on most operating systems, with no hypervisor involvement. Web browsers and operating systems do not consider such denial of service attacks to be security vulnerabilities and do not issue CVEs for similar behavior.

2.8.1.2 VMSA-2019-0005

CVE-2019-5518, CVE-2019-5519
These out-of-bounds writes and TOCTOU issues in the UHCI controller (USB1.0) apply to ESXi. They were fixed in ESXi670-201903001. Thus, the issues are fixed in this release.
CVE-2019-5514
This API authentication issue does not apply to ESXi.
CVE-2019-5515, CVE-2019-5524
These out-of-bounds writes in the e1000 virtual network adapter do not apply to ESXi.

2.8.1.3 VMSA-2019-0008

This issue covers Microarchitectural Data Sampling (MDS) Vulnerabilities, a category of Spectre issue. This issue is further subdivided into “sequential” attacks and “concurrent” attacks. The “sequential” category is fully fixed by the evaluated patch release. The “concurrent” category can only be mitigated at a performance penalty. VMware KB 67577 covers enabling the ESXi Side-Channel-Aware Scheduler to mitigate the “concurrent” category, at a potential performance penalty. The evaluated patch release includes ‘v2’ of the ESXi Side-Channel-Aware Scheduler as described in the KB article, which is disabled by default due to the performance penalty. The KB article describes using the ESXi Host Client to enable the scheduler, or use esxcli:
esxcli system settings kernel set -s hyperthreadingMitigation -v TRUE
esxcli system settings kernel set -s hyperthreadingMitigationIntraVM -v FALSE

2.8.1.4 VMSA-2019-0011

CVE-2019-5528
This issue is a partial denial of service vulnerability in the hostd process. The issue is present in the evaluated patch release. An administrator has two options:
1) Update to ESXi670-201908201-UG, which was released following the evaluated patch release.

2) Mitigate by making a configuration change using SSH as described in the VMSA. VMware KB 67920 covers applying this mitigation.

2.8.1.5 VMSA-2019-2012

CVE-2019-5521, CVE-2019-5684
These issues are out-of-bounds read/write issues in virtual 3D graphics. The issues were fixed in ESXi670-201904101, a predecessor to the evaluated patch release. Thus, the issues are fixed in this release.

2.8.1.6 VMSA-2019-0013

CVE-2017-16544
This issue is a command injection vulnerability due to un-sanitized filenames in the ESXi shell. The issues were fixed in ESXi670-201904101, a predecessor to the evaluated patch release. Thus, the issues are fixed in this release.
CVE-2019-5531
This issue is an information disclosure vulnerability in the ESXi Host Client. The issue was fixed in ESXi670-201810101 (patch in October 2018), a predecessor to 6.7 Update 2. Thus, the issue is fixed in this release.

2.8.1.7 VMSA-2019-0014

CVE-2019-5527
This issue is a use-after-free in the virtual sound device. The issue was fixed in ESXi670-201904101, a predecessor to the evaluated patch release. Thus, the issue is fixed in this release.

2.8.1.8 VMSA-2019-0019

CVE-2019-5536
This issue is a guest denial-of-service in the virtual 3D graphics device, which allows an unprivileged guest shader to crash the guest. The issue is present in the evaluated patch release. An administrator has two options:

1) Update to ESXi670-201908101-SG, which was released following the evaluated patch release.

2) Mitigate by removing the virtual 3D graphics device, as discussed for VMSA-2018-0025 above.

2.8.2 OpenSSH
These issues all relate to the usage of ‘scp’ (secure copy) as a client – for example, they would apply if invoking scp from the ESXi shell to copy bits from an external server to ESXi. The ‘scp’ binary is included with ESXi for troubleshooting purposes and is not intended for production use.

CVE-2018-20685: VMware included a fix in the openssh version 7.9 included in ESXi 6.7 Update 2 and later. This issue is fixed in the evaluated patch release.
CVE-2019-6109, -6110, -6111: OpenSSH has not yet released fixes for these issues and considers them usability (non-security) issues.

Customers who wish to mitigate the risk of unintentional use of ‘scp’ may disable outbound SSH access (which includes the ‘scp’ command). To do so, execute the following command to disable the outbound firewall rule:

```bash
esxcli network firewall ruleset set -e no -r sshClient
```

As shell and inbound SSH access are already disabled by default for ESXi, VMware has no immediate plans to explicitly block outbound SSH access.
3 OPERATIONAL GUIDANCE: ADVANCED OPTIONS

Advanced settings are specific key/value configuration settings that can be set to manipulate various settings. For more information about configuring advanced options, see the VMware knowledge base article at https://kb.vmware.com/s/article/1038578.

3.1 Configuring Advanced Options Using the UI

The ESXi Host Client (UI) contains a tab which allows directly editing advanced options. This is also the best approach for discovering which options exist and what their behaviors are. For more information about using the Host Client to configure advanced options, see the following topic: https://docs.vmware.com/en/VMware-vSphere/6.7/com.vmware.vsphere.html#hostclient.doc/GUID-CADD24CB-207C-440C-92DD-A9E74305281C.html

3.2 Configuring Advanced Options Using the VIM API

Setting advanced options on individual hosts using the UI can be impractical at scale. The VIM API offers a programmatic interface to manipulate advanced options. The UI is built on top of this VIM API interface. Advanced options are controlled using the OptionManager managed object. This managed object can be accessed from the HostSystem managed object’s ConfigManager field, as HostSystem.configManager.advancedOption. For more information about using this managed object, see the following topics: https://code.vmware.com/apis/358/vsphere#/doc/vim.HostSystem.html https://code.vmware.com/apis/358/vsphere#/doc/vim.host.ConfigManager.html https://code.vmware.com/apis/358/vsphere#/doc/vim.option.OptionManager.html

3.2.1 Querying Advanced Options
The full list of advanced options is available using the OptionManager.supportedOption[] field. The contents of this array are fixed for a particular ESXi build and do not change at runtime. The list of currently set advanced options (for example, those set to non-default values) is available using the OptionManager.setting[] field. The OptionManager.UpdateOptions method can be used to search for option keys matching a particular string, as the full list of advanced options might be too unwieldy for interactive use.

3.2.2 Setting Advanced Options
To set an advanced option, invoke the OptionManager.UpdateOptions method with the desired key and value.
The following pseudocode sets the advanced option `Config.Etc.issue` to “Hello World\n”:

MoRef optionMgr = hostSystem.configManager.advancedOption;
opts = new OptionValue[]
opts[0].key = “Config.Etc.issue”
opts[0].value = “Hello World\n”
optionMgr.UpdateOptions(changedValue=opts)

### 3.3 Selected Advanced Options

A full list of advanced options is impossible to document. Options can be added or removed in any release. The best reference for the advanced options a release supports is the UI or API exposed by that release, which includes a description of any advanced options.

The following table describes advanced options which have a particular security effect.

<table>
<thead>
<tr>
<th>Option name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotations.WelcomeMessage</td>
<td>(empty)</td>
<td>Displayed in Host Client (UI) prior to login. Displayed in DCUI on default screen. <em>(Note: In DCUI, replaces some display text like host IP address.)</em></td>
</tr>
<tr>
<td>Config.Etc.Issue</td>
<td>(empty)</td>
<td>Displayed by SSHD prior to login. <em>(Note: Trailing newline recommended.)</em></td>
</tr>
<tr>
<td>Config.Etc.Motd</td>
<td>(empty)</td>
<td>Displayed by SSHD following login.</td>
</tr>
<tr>
<td>Config.HostAgent.vmacore.soap.sessionTimeout</td>
<td>30 (minutes)</td>
<td>Idle time in minutes before VIM API session is automatically logged out. 0 disables. Applies to new sessions only. <em>(Note: New in 6.7 Update 2.)</em></td>
</tr>
<tr>
<td>Mem.MemEagerZero</td>
<td>0 (disabled)</td>
<td>Enable zeroing userworld and guest memory pages (including VMM) after VM exit. <em>(Note: New in 6.7 Update 2.)</em></td>
</tr>
<tr>
<td>Security.AccountLockFailures</td>
<td>5</td>
<td>Maximum number of failed login attempts before a user's account is locked. Zero disables account locking. <em>(Note: New in 6.7 Update 2.)</em></td>
</tr>
<tr>
<td>Security.AccountUnlockTime</td>
<td>900</td>
<td>Number of seconds that a user is locked out.</td>
</tr>
<tr>
<td>Security.PasswordHistory</td>
<td>0</td>
<td>Number of passwords to remember for each user. Prevents duplicate or similar passwords.</td>
</tr>
<tr>
<td>Security.PasswordMaxDays</td>
<td>99999</td>
<td>Maximum number of days between password changes.</td>
</tr>
<tr>
<td>Security.PasswordQuality Control</td>
<td>retry=3, min=disable</td>
<td>Pam_passwordqc configuration.</td>
</tr>
<tr>
<td>UserVars attribute</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>UserVars.DcuiTimeOut</td>
<td>600 (10 minutes)</td>
<td>Idle time in seconds before DCUI is automatically logged out. 0 disables.</td>
</tr>
<tr>
<td>UserVars.ESXiShellInteractiveTimeOut</td>
<td>0</td>
<td>Idle time in seconds before an interactive shell is automatically logged out. Takes effect for new sessions only. 0 disables.</td>
</tr>
<tr>
<td>UserVars.ESXiShellTimeOut</td>
<td>0</td>
<td>Time in seconds a login shell will wait for login. 0 disables.</td>
</tr>
<tr>
<td>UserVars.HostClientSessionTimeout</td>
<td>900 (15 minutes)</td>
<td>Idle time in seconds before Host Client is automatically logged out. 0 disables.</td>
</tr>
<tr>
<td>UserVars.HostClientWelcomeMessage</td>
<td>(empty)</td>
<td>Displayed in Host Client following login, as a “hint”.</td>
</tr>
</tbody>
</table>

Table 2: Advanced Options
4 OPERATIONAL PROCEDURES FOR ADMINISTRATORS

This section describes additional steps, clarifications, and exclusions that might not be documented in the public documentation for this product. The assumption is that the TOE and its environment has already been successfully set up and working before these next steps are performed.

4.1 Audit Configuration (FAU)

ESXi offers both local and remote audit recordkeeping. This is disabled by default and must be manually enabled for both local and remote modes. The local audit log operates as a fixed-size buffer of recent audit messages. Once filled, new records will overwrite the oldest records. The remote audit log forwards the same stream of audit records in a standard syslog format (RFC 3164 “The BSD syslog protocol”) to a remote server, either unencrypted or encrypted (RFC 5425 “TLS Transport Mapping for Syslog”). Audit messages comply with RFC 5424 “The Syslog Protocol”, but general syslog messages only comply with RFC 3164. A generated audit message is sent simultaneously to the local store and remote store. If the connection to the remote store is lost, any generated audit messages will be dropped from the perspective of the remote store. Upon reconnection, an audit message is generated indicating potential message loss.

4.1.1 Viewing Audit Records
Audit records can be viewed remotely using a remote audit server (see section 4.1.3) or locally using the ESXi application /bin/auditLogReader. This application can be invoked using SSH access.

4.1.2 Configuring Local Audit Records
Configuration of local auditing is through ESXCLI, in the esxcli system auditrecords namespace. The only parameter of note is an audit record size (in kilobytes). 4 MB (4096 kilobytes) is a recommended size.
Example command sequence:
```
esxcli system auditrecords local set --size=<size-in-kb>
esxcli system auditrecords local enable
```
For more information about ESXCLI commands, see the following topic: https://code.vmware.com/docs/6676/vsphere-command-line-interface-reference#doc/esxcli_system.html
These commands are new as of ESXi 6.7 Update 2.

4.1.3 Configuring Remote Audit Server
Configuration of the remote audit server is through ESXCLI, in the esxcli system syslog namespace. Audit events are passed in a format similar to RFC 3164 and RFC 5424. Audit records have the RFC-specified prefix <110>, which is a packed value reflecting audit
facility 13 (for “audit”) at the “info” level. The firewall must be configured to permit outbound syslog access.
Communication is through TLS. A CA root suitable for the syslog server must be loaded into the ESXi CA root. For more information, see section 4.4.4. Syslog can be used on port 514 or 1514. When used with TLS (recommended), the preferred port is 1514. Prefix with \tcp:// for unencrypted syslog, and with \ssl:// for encrypted syslog.
Multiple syslog servers can be specified, separated by commas.
The following example command sequence configures a firewall to permit outbound syslog:
esxcli network firewall ruleset set --ruleset-id=syslog --enabled=true
esxcli network firewall refresh
The following example command enables remote audit logging:
esxcli system auditrecords remote enable
The following example command sequence sets the syslog server to syslog.example.com:
esxcli system syslog
    config set --loghost="ssl://syslog.example.com:port"
esxcli system syslog reload
The following example command sequence clears syslog configuration:
esxcli system syslog config set --reset=loghost
esxcli system syslog reload
The following example command sequence enables X.509 and CRL checking.
esxcli system syslog config set --crl-check="true"
esxcli system syslog config set --x509-strict="true"
esxcli system syslog reload
For more information about ESXCLI commands, see the following topics:
The “crl-check” option enables verification of X.509 CRLs, which are not checked by default in compliance with industry convention. The NIAP verified configuration, however, requires CRL checks. Due to implementation limitations, if CRL checks are enabled then all certificates in a certificate chain must provide a CRL link.
The “x509-strict” option performs additional validity checks on CA root certificates during verification. These checks are generally not performed (CA roots are inherently trusted) and might cause incompatibilities with existing, misconfigured CA roots. The NIAP requirements, however, require even CA roots to pass validations.
VMware does not recommend enabling the “crl-check” or “x509-strict” options because of the difficulty in properly configuring an environment that uses CRL checks. In addition, the vCenter Server Certificate Authority (VMCA) does not support CRLs and is outside the scope of this evaluation.

4.2 Configuration (FCS)

4.2.1 Cryptographic Key Generation
ESXi generates several asymmetric keys used during normal operation.
4.2.1.1 TLS (SSL) Key

The Transport Layer Security (TLS) key is used to secure communication with the host using the TLS protocol. The TLS key is generated at first system boot as a 2048-bit RSA key. ESXi does not currently implement automatic generation of ECDSA keys for TLS. The TLS private key is not intended to be serviced by the administrator.

The TLS key is stored on the file system at:

/etc/vmware/ssl/rui.key

The TLS public key (including intermediate CAs) is stored on the file system as an x509v3 certificate at:

/etc/vmware/ssl/rui.crt

The default TLS certificate is self-signed, with a subjectAltName field matching the host name at installation. The administrator might want to install a different certificate, for example to make use of a different subjectAltName or to include a particular Certificate Authority (CA) in the verification chain.

The VIM API HostCertificateManager (vim.host.CertificateManager) managed object offers methods to manage the certificate. A typical workflow would call GenerateCertificateSigningRequest to fetch a Certificate Signing Request (CSR), use an external mechanism (beyond the scope of this document) to create a signed certificate, then would call InstallServerCertificate to install the signed certificate. For more information, see the following topic:


If using ESXi in conjunction with vCenter Server, vCenter Server will automatically generate a CSR and sign it using the vCenter Managed Certificate Authority (VMCA), and install the resulting certificate, when adding an ESXi host to vCenter Server. Usage of vCenter Server is beyond the scope of this guidance document.

The Host Client UI also offers an interface for changing the TLS certificate. For more information, see the following topic:


ESXi’s CGI interface also offers a mechanism to directly set these using HTTPS PUT to the URI /host/ssl_cert and /host/ssl_key. See section 5.5.1. For more information, see the following topic:


4.2.1.2 SSH Key

The SSH key secures communication with the host using the SSH protocol. The SSH key is generated when the system first boots as a 2048-bit RSA key. The SSH server is disabled by default but can be enabled through the VIM API or the physical console (DCUI). SSH access is intended primarily for troubleshooting purposes; logging in through SSH requires administrative privileges equivalent to full host control.

The SSH public and private keys are stored at the following locations:

/etc/ssh/ssh_host_rsa_key

ESXi’s CGI interface also offers a mechanism to directly set these through HTTPS PUT to the URI `/host/ssh_host_rsa_key` and `/host/ssh_host_rsa_key_pub`. See section 5.5.1.

The SSH key is not intended to be serviced by the administrator.

### 4.2.2 Cryptographic Key Establishment

#### 4.2.2.1 TLS Cryptographic Key Establishment

Configuration of TLS cryptographic key establishment is governed by choice of TLS cipher suites, which will select one of the RSA-based key transports (as specified in SP 800-56B) or ECC-based key agreements using ephemeral Ecliptic Curve Diffie Hellman (ECDH) (as specified in SP 800-56A). Choice of TLS cipher suites is covered in section 4.2.9 “TLS Protocol”.

#### 4.2.2.2 SSH Cryptographic Key Establishment

Configuration of SSH cryptographic key establishment is governed by the SSHD configuration. ESXi ships with a default configuration that permits RSA-based key transport (as specified in SP 800-56B), ephemeral Diffie Hellman (DH) (as specified in SP 800-56A) key agreement, and ephemeral Ecliptic Curve Diffie Hellman (ECDH) (as specified in SP 800-56A). The SSHD configuration is not intended to be serviced by the administrator. Choice of SSH configuration is covered in section 4.2.8 “SSH”.

### 4.2.3 Cryptographic Key Destruction

#### 4.2.3.1 Volatile Key Destruction

An advanced option `MemEagerZero` can be used to enable zeroing out userworld and guest memory pages (including VMM) when userworlds and guests exit. See section 3 for further discussion of advanced options.

```
Mem.EagerZero=1
```

This advanced option can also be set using ESXCLI:

```
esxcli system settings advanced set -o "/Mem/MemEagerZero" --int-value "1"
```

For more information about ESXCLI commands, see the following topic:


### 4.2.4 Cryptographic Operation

#### 4.2.4.1 FIPS Mode

ESXi defaults to using FIPS-validates (CAVP information in section 5.2.1) cryptography for all operations needing cryptography for security purposes. This includes remote access to
management interfaces (for example, TLS and SSH), key generation, signature verification, and other use cases.
Due to the nature of the ESXi product where all management functions are remote and require cryptography, failing a FIPS self-test would render the product unusable and unrecoverable short of re-installation. Thus, ESXi implements several “FIPS mode” switches to disable enforcement of the FIPS self-tests in certain subsystems. These switches can be examined and set using ESXCLI.
To get, enable, or disable “FIPS mode” for TLS on the HTTPS interface:

```
esxcli system security fips140 rhttpproxy get
esxcli system security fips140 rhttpproxy set -e true
esxcli system security fips140 rhttpproxy set -e false
```

To get, enable, or disable “FIPS mode” for SSH:

```
esxcli system security fips140 ssh get
esxcli system security fips140 ssh set -e true
esxcli system security fips140 ssh set -e false
```

For more information about ESXCLI commands, see the following topic: https://code.vmware.com/docs/6676/vsphere-command-line-interface-reference/#/doc/esxcli_system.html

There are no expected modes which would lead to a failure to pass FIPS self-tests. In practice, failure would only occur in the event of hardware failure, for example random bit errors in memory.
Other use cases (for example, key generation) always use FIPS-validated cryptography and are not configurable.

4.2.4.2 Other Parameters

There are no other configurable parameters relating to cryptographic operation.

4.2.5 Random Bit Generation

There are no configurable parameters relating to random bit generation.

4.2.6 Entropy for Virtual Machines

There are no configurable parameters relating to entropy for virtual machines.

4.2.7 HTTPS

There are no configurable parameters relating to usage of HTTPS.

4.2.8 SSH
Parameters used by the SSH server are not intended to be serviced by the administrator. However, the default configuration is not compliant with the SSH Protection Profile as specified in the Security Target (ST) so the administrator will need to modify the system for compliance.
As noted below, the differences are compliant with FIPS validation and not expected to alter security.
Most SSH parameters are as defined by OpenSSH, and are stored in the following file:

```
/etc/ssh/sshd_config
```

This file is not intended to be modified by the administrator. Manual modification can result in errors as the system automatically makes certain edits to this file.
To enable SSH access through VIM, use VIM’s HostSystemService (vim.hostSystemService) managed object. Find the SSH service in HostSystemService.serviceInfo.services[], and supply the corresponding key to HostSystemService.Start(). For more information, see the following topic:

To enable SSH access through the ESXi console (DCUI), follow the instructions in VMware ESXi Installation and Setup documentation:

The Host Client UI also has an interface for enabling the SSH server. See the following topic:
https://docs.vmware.com/en/VMware-vSphere/6.7/com.vmware.vsphere.html.hostclient.doc/GUID-B649CB74-832F-467B-B6A4-8BA67AD5C1F0.html

### 4.2.8.1 Setting Up Public Key Authentication

ESXi implements public key authentication by allowing public keys. The file that contains permitted public keys is initially empty. It is stored at the following location:

```
/etc/ssh/keys-root/authorized_keys
```

To configure the host for public key access, use standard OpenSSH syntax to append the public key to the authorized_keys file. You must log in to the host using SSH and authenticate with a valid user name and password.

The following example command installs a public key for SSH access. When executed, the command prompts for the root password of the ESXi host where it is run.

```
cat ~/.ssh/id_rsa.pub | \\
    ssh root@<esxhost> "cat >> /etc/ssh/keys-root/authorized_keys"
```

When configured with this public key, the ESXi host will allow any SSH connection which can prove knowledge of the private key to authenticate as user “root”.

ESXi’s CGI interface also offers a mechanism to directly set the authorized_keys file through HTTPS PUT to the URI /host/ssh_root_authorized_keys. See section 5.5.1.

### 4.2.8.2 Selecting Ciphers

Ciphers are selected in the sshd_config file. The default list of ciphers is:

```
Ciphers aes256-gcm@openssh.com,aes128-gcm@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr
```

The NIAP SSH Protection Profile does not recognize aes192-ctr as a valid cipher, so administrators are advised to use the following cipher list instead:

```
Ciphers aes256-gcm@openssh.com,aes128-gcm@openssh.com,aes256-ctr,aes128-ctr
```

**Note:** VMware has FIPS-validated aes192-ctr and believes its omission from the NIAP protection profile to be unintentional.
4.2.8.3 Selecting Authentication Mechanism

Authentication mechanisms are selected in the sshd_config file. The default list of authentication mechanisms is:

```
PubkeyAcceptedKeyTypes ecdsa-sha2-nistp256-cert-v01@openssh.com,ecdsa-sha2-nistp384-cert-v01@openssh.com,ecdsa-sha2-nistp521-cert-v01@openssh.com,ssh-rsa-cert-v01@openssh.com,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,rsa-sha2-512,rsa-sha2-256,ssh-rsa
```

The NIAP SSH Protection Profile does not permit the various -cert-v01@openssh.com ciphers (used to permit CA-based public key login), so administrators are advised to use the following cipher list instead:

```
PubkeyAcceptedKeyTypes ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,rsa-sha2-512,rsa-sha2-256,ssh-rsa
```

**Note:** No instructions are available to configure the CA-based public key login key types. The removal is to satisfy SSH Protection Profile requirements that no additional key types be supported.

**Note:** ESXi will generate a DSA key for historical reasons. However, the standard SSHD configuration will not permit this DSA key to be used.

4.2.8.4 Selecting MAC

MAC algorithms are selected in the sshd_config file. The default list of MAC algorithms is:

```
MACs hmac-sha2-256,hmac-sha2-512,hmac-sha1
```

4.2.8.5 Selecting Key Exchange

Key exchange algorithms are selected in the sshd_config file. The default list of key exchange algorithms is:

```
```

The NIAP Extended Package for SSH does not permit strong SSH Diffie-Hellman groups defined by RFC 4419 (diffie-hellman-group-exchange-sha256) and RFC 8268 (groups 14, 16, and 18), so administrators are advised to use the following cipher list instead:

```
KexAlgorithms ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-group14-sha1
```

**Note:** The RFC 8268 Diffie-Hellman groups are included as “Approved IKE groups for FFC key agreement” in SP 800-56Arev3 and are thus usable for FIPS-validated key exchange. It is hoped a future reversion of the Extended Package for SSH will include these groups.

4.2.9 TLS Protocol

ESXi uses a shared configuration for TLS Client and TLS Server protocols, though some TLS clients (including remote syslog) are not configurable. The default configuration is compliant with the TLS requirements in the Virtualization Protection Profile as specified in the Security Target (ST). These settings are not generally expected to be serviced by an administrator, though
in practice public reports of TLS security vulnerabilities and company best-practices have necessitated manual configuration. VMware supports altering these settings only as directed by Global Support Services (GSS) or using the TLS Configurator Utility.

4.2.9.1 TLS Protocol Versions

By default, ESXi allows TLS v1.2 only. ESXi supports configuring to enable TLS v1.1, TLS v1.0, and SSL v3 to support legacy clients and installations; this is not recommended and doing so is explicitly non-compliant with the Security Target. TLS v1.3 is not yet supported by ESXi. The default protocol list is:
"tls1.2"
A maximally-permissive (and insecure) protocol list would be:
"tls1.2,tls1.1,tls1.0,ssl3"
ESXi does not support “holes” in the protocol list.

4.2.9.2 TLS Cipher Suites

The default ESXi cipher suite permits a selection of high security protocols, with first preference to ciphers implementing GCM for better performance, and second preference to ciphers which use ephemeral key agreement (ECDHE) for Perfect Forward Secrecy. Static key ciphers (RSA) are maintained for compatibility with the TLS1.2 standard (which makes RSA+AES mandatory). ESXi does not use ECDSA server certificates, so ECDSA suites are only used as a client configuration.
The cipher suite specification is as documented by OpenSSL. The default cipher list is:
"ECDHE+AESGCM:RSA+AESGCM:ECDHE+AES:RSA+AES"
This results in the following ciphers, in preference order:
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA
4.2.9.3  **ECC Choices in TLS**

ESXi defaults to a selection of strong prime curves. "prime256v1:secp384r1:secp521r1"

4.2.9.4  **Configuration Files**

<table>
<thead>
<tr>
<th>External Interface</th>
<th>Configuration File</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTPS (443)</td>
<td>/etc/vmware/rhttproxy/config.xml</td>
<td><code>&lt;config&gt;&lt;vmacore&gt;&lt;ssl&gt;</code></td>
</tr>
<tr>
<td>NFC file transfers (902)</td>
<td>/etc/vmware/hostd/config.xml</td>
<td><code>&lt;cipherList&gt;ECDHE+AESGCM:…&lt;/cipherList&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;curvesList&gt;prime256v1:…&lt;/curvesList&gt;</code></td>
</tr>
<tr>
<td>Authd (902)</td>
<td>/etc/vmware/config</td>
<td><code>tls.ciphers = “ECDHE+AESGCM:…”</code></td>
</tr>
<tr>
<td>MKS (902)</td>
<td></td>
<td><code>tls.curves = “prime256v1,…”</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>tls.protocols = “tls1.2”</code></td>
</tr>
</tbody>
</table>

Table 3: TLS Settings in Configuration Files

4.2.9.5  **The TLS Configurator Utility**

When deployed in conjunction with vCenter Server, the TLS Configurator Utility can be used to automate the setting of TLS protocol versions across an entire datacenter. The TLS Configurator Utility does not support the setting of TLS cipher suites. Usage of the TLS Configurator Utility is beyond the scope of this document; the implementation consists of a script running within vCenter Server which automates the settings previously described. The TLS Configurator Utility is documented in the vSphere Security documentation: https://docs.vmware.com/en/VMware-vSphere/6.7/com.vmware.vsphere.security.doc/GUID-82028A21-8AB5-4E2E-90B8-A01D1FAD77B1.html

4.3  **Protection of User (VM) Data (FDP)**

4.3.1  **Virtual Networking**

Networking on an ESXi system is configured through the VIM API or ESXCLI. The VIM API is the primary interface and ESXCLI is used primarily for diagnostics or support.

4.3.1.1  **Virtual Networking (Host) - VIM API**

The primary VIM API entry point for configuring networking is the `vim.Host.NetworkSystem` managed object. This interface is used to configure port groups, virtual switches, and connections to the physical network. In the context of this API, a “Virtual NIC” is a connection between the host network stack and virtual switches; NICs connected to virtual machines are configured separately.
In a typical deployment, vCenter Server or NSX uses these APIs to manage the network configuration at datacenter scope; direct user control of network configuration is uncommon. The vCenter Server and NSX components are outside the scope of this evaluation. For more information, see the following topic:

### 4.3.1.2 Virtual Networking (Host) - ESXCLI

Commands to configure networking on the ESXi host are in the esxcli network namespace. For more information, see the following topic:

### 4.3.1.3 Virtual Networking (Host) – Host Client UI

The Host Client UI offers an interface to configure the host’s port groups, virtual switches, and physical connections. For more information, see the following topic:

### 4.3.1.4 Virtual Networking (Virtual Machines)

The configuration for virtual networking is through the VIM API. A virtual machine managed object’s configuration (vim.VirtualMachine.config) contains an array of devices (vim.vm.ConfigInfo.hardware.device[]). Within that array, some devices are network devices (classes derived from vim.vm.device.VirtualEthernetCard); these devices will have a backing property that contains the configured virtual network.

<table>
<thead>
<tr>
<th>Type of Backing Object</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>vim.vm.device.VirtualEthernetCard.DistributedVirtualPortBackingInfo</code></td>
<td>Connected to a Distributed Virtual Switch (built-in virtual network switch).</td>
</tr>
<tr>
<td><code>vim.vm.device.VirtualEthernetCard.OpaqueNetworkBackingInfo</code></td>
<td>Connected to a virtual network that is managed by NSX. (Not covered by this guide).</td>
</tr>
<tr>
<td><code>vim.vm.device.VirtualSriovEthernetCard.SriovBackingInfo</code></td>
<td>Connected to a passthrough network device. The</td>
</tr>
</tbody>
</table>
Virtual machines can communicate if connected to the same Distributed Virtual Switch (or if the Distributed Virtual Switches are connected, either directly or indirectly); if connected to the same NSX Opaque Network (in which case connectivity is managed by NSX); or if connected to an SR-IOV passthrough device and the passthrough device is on the same physical network. 

**Note:** NSX and SR-IOV devices are not available in the TOE. NSX is a separate product, and SR-IOV is a special form of PCI Passthrough device.

When using Distributed Virtual Switches, the virtual machine can only connect to the host’s network address if the host’s vmknic is connected to the same Distributed Virtual Switch.

To disable access to a virtual switch, either remove the virtual NIC from the virtual machine, or configure the virtual NIC to be “disconnected” (have no backing object).

The Host Client UI offers an interface to configure the virtual machine’s virtual NIC and network connections.

For more information, see the following topic:


### 4.3.2 Physical Platform Resources

ESXi allows direct access to physical devices in limited scenarios. Most devices are implemented as virtual devices. Access attempts from the guest are trapped and implemented in software. (For example, the keyboard is implemented by trapping IO port access in the guest and feeding keycode data obtained using a remote console connection; it is not possible to connect the physical keyboard to any VM). Several broad categories of device can be configured for direct access, including USB, Storage, physical CD-ROM or DVD, vGPU, and PCI Passthrough (“DirectPath I/O”) devices. Physical device access can be revoked by removing the associated virtual device from the virtual machine configuration.

Use of physical (“passthrough”) resources is uncommon, as doing so prevents vMotion.

#### 4.3.2.1 USB

The virtual machine’s ConfigInfo contains an array of device objects (vim.vm.ConfigInfo.hardware.devices[]). Some instances in that array can be of type vim.vm.device.VirtualUSB; these devices can have a backing field, which specifies access to a particular backing device. Note: For connectivity, the virtual machine must also have a compatible virtual USB Controller configured for the virtual machine.

<table>
<thead>
<tr>
<th>Type of Backing Object</th>
<th>Notes</th>
</tr>
</thead>
</table>

Table 4: BackingInfo Types for Virtual Network Devices

corresponding device must be configured for passthrough on the host. (This is not a common configuration.)
vim.vm.device.VirtualUSB.USBBackingInfo | USB device attached to the ESXi host. (Disallows vMotion).
--- | ---
vim.vm.device.VirtualUSB.RemoteHostBackingInfo | USB device attached to a particular remote ESXi host. (Used to keep device attached during vMotion.)
vim.vm.device.VirtualUSB.RemoteClientBackingInfo | USB device forwarded over the network using the VM’s remote console.

| Table 5: BackingInfo Types for Virtual USB Devices |
|---|---|

For more information, see the following topic: https://code.vmware.com/apis/358/vsphere#/doc/vim.vm.device.VirtualUSB.html

Configuring USB devices attached to the host available for connection to a virtual machine is via ESXCLI. The following command prints available devices, including VID (Vendor ID) and PID (Product ID) used to configure USBBackingInfo as described above:

```
esxcli hardware usb passthrough device list
```

Configuring USB devices attached to a virtual machine is possible using the Host Client. The procedure is:

1. Click **Virtual Machines** in the VMware Host Client inventory.
2. Right-click a virtual machine in the list and select **Edit settings** from the pop-up menu.
3. On the **Virtual Hardware** tab, select **Add other device** and select **USB device** from the drop-down menu. Note: this option may be disabled if the virtual machine lacks a **USB controller**, which can be added using the same procedure. This option may also be disabled if the host has no available USB devices to add.

The USB device appears in the Virtual Hardware devices list.

4. To delete an existing hard disk, move your pointer over the disk and click the **Remove** icon (X).

4.3.2.2 **Storage**

Passthrough storage devices are known as Raw Device Mappings (RDM) and represent essentially a passthrough of SCSI commands to a particular LUN. Configuring an RDM occurs through the `vim.vm.ConfigInfo.hardware.devices[]` array, with instances of `type vim.vm.device.VirtualDisk` that have a backing field of `type vim.vm.device.VirtualDisk.RawDiskVer2BackingInfo`.

4.3.2.3 **Physical CD-ROM/DVD**

A physical CD-ROM/DVD drive on a host can be attached directly to a virtual machine. Such drives are automatically detected by the host; no configuration is required. Section 4.6.3.3 describes configuring a virtual machine to have access to a physical CD-ROM or DVD drive.
4.3.2.4 vGPU

When used in conjunction with an nVidia graphics card and the nVidia GRID software package, a vGPU can be passed through to one or more virtual machines for GPU/GPGPU acceleration. Due to the lack of hardware isolation between the virtual machines, this configuration is not allowed under NIAP.

4.3.2.5 PCI Passthrough

ESXi allows specific PCI devices to be configured for passthrough usage by a VM instead of for usage by the host; this feature is also known as “DirectPath I/O” and is used in limited scenarios for improved low-latency performance. When operating in this configuration, the PCI device is required to be behind a VT-d bridge which implements an IOMMU; vmkernel arranges for this IOMMU to be programmed with only pages which the guest is allowed to access, which ensures the guest can only program the PCI device to access pages to which the guest has access. If the virtual machine powers off or becomes unavailable, the PCI device will receive a PCI “reset” to clear any pending DMA transactions before detaching the IOMMU domain from the device. The host configuration that indicates which devices are eligible for assignment to virtual machines is through the vim.host.PciPassthruSystem managed object (accessible using the host’s ConfigManager.pciPassthruSystem). The UpdatePassthruConfig method on this managed object can be used to enable or disable PCI passthrough for a particular PCI bus address (bus:slot:function). Making a device eligible for passthrough generally requires a host reboot. Devices not configured for PCI passthrough on the host are instead claimed by ESXi and are thus unavailable to virtual machines.

The VM configuration uses vim.vm.ConfigInfo.hardware.devices[] again. When an instance in that array is of type vim.vm.device.VirtualPCIPassthrough, the backing field will have type vim.vm.device.VirtualPCIPassthrough.DeviceBackingInfo and this object will describe the exact PCI device which is being passed through. If a VM specifies a PCI device which has not been configured for assignment to VMs by the administrator, the VM will be unable to power on.

4.3.2.6 Physical Network

ESXi does not allow a guest to have direct access to a physical network except through the administrator configuring a virtual switch with a network interface attached to specific physical network interfaces. Any virtual machines expected to have external network access may be configured to have their virtual network interfaces connected to this virtual switch. See section 4.3.1 for information about configuring virtual switches and virtual network interfaces and their backings.
More commonly, the administrator can configure a virtual switch to connect to a specific VLAN on the physical network and then connect virtual machines to that virtual switch. This provides logical isolation of the virtual machines from the physical network. See section 5.8.1 for further information.
4.3.3 Memory Zeroization

4.3.3.1 Volatile Memory

There are no configurable parameters related to volatile memory zeroization.

4.3.3.2 Non-volatile Memory

There are no configurable parameters related to non-volatile memory zeroization.

4.3.4 Hardware-based VM Isolation

Prior ESXi releases supported binary translation and software page tables. This functionality was removed in ESXi 6.7. The only supported virtualization mode in ESXi 6.7 uses hardware-based virtualization (VT-x or AMD-V, as the platform’s CPU supports) in conjunction with hardware-based two-level page tables (Extended Page Tables).

*Note: the TOE is evaluated only on Intel CPUs with VT-x instructions with EPT. Thus, AMD-V is not included in this evaluation.*

Usage of hardware-based VM isolation mechanisms is always enabled. For more information, see the following topic:


4.4 Authentication Configuration (FIA)

4.4.1 Authentication Failure Handling

Authentication failure handling is available through several advanced options. See the following options in section 3:

- Security.AccountLockFailures
- Security.AccountUnlockTime

The `AccountLockFailures` option indicates the permitted number of failed login attempts before locking the account. For example, to lock the account on the 5\(^{th}\) login failure, set this value to 4. For implementation reasons, some login mechanisms count unexpectedly:

- VIM logins (including Host Client) and ESXCLI reflect the exact number of failed logins.
- SSH connections count as a login attempt when displaying a password prompt, and undo that count on successful login. This is normal challenge/response behavior.
- CGI logins double-count login failures. This is an implementation error in ESXi 6.7, which will be corrected in a future software version; it does not affect security.

*Warning: due to this error, a user may be locked out faster than the number of failed logins using the CGI interface.*

The `AccountUnlockTime` option describes a time interval following the last failed login attempt before a successful login for a given account is allowed. Any login attempt within the lock timeout will restart the lock timeout.
### 4.4.2 Password Management

Password management is available through several Advanced Options. See the following options in section 3:

- Security.PasswordQualityControl
- Security.PasswordHistory
- Security.PasswordMaxDays

The PasswordQualityControl setting is a set of *passwdqc* options (search online for “man passwdqc.conf” for a description of this standard module). Special characters are allowed. Lengths of at least 15 characters are supported. The setting consists of options separated by a space. Selected options include:

- `min=N0,N1,N2,N3,N4` where N0 is minimum length of passwords from a single character class, N1 is minimum length of passwords from two character classes, N2 is minimum length for a passphrase, N3 is minimum length for three character classes, and N4 is minimum length for four character classes. “disabled” may be used to disallow passwords with the specified number of character classes. The four character classes are: digits, upper case letters, lower case letters, other.

- `max=N` where N is the maximum allowed password length.

- `similar=permit|deny` whether a password is allowed to be similar to the old password. Passwords are evaluated by complexity after removing any long common substring.

The default PasswordQualityControl setting requires three character classes and a minimum length of seven characters:

```
min=disabled,disabled,disabled,7,7
```

If implementing the DoD Annex, the `similar=deny` option plus a minimum password length can be combined to enforce a requirement that passwords are sufficiently different. The password history setting is only enforced for passwords changed through VIM API’s `LocalAccountManager.changePassword`, so the administrator is required to use that interface to change the password. The following PasswordQualityControl setting, in conjunction with a PasswordMaxDays setting, satisfies the requirements of the DoD Annex:

```
min=disabled,disabled,disabled,disabled,15 similar=deny
```

The PasswordHistory and PasswordMaxDays settings are new in ESXi 6.7 Update 2. As an example, to set the password complexity requirement to require 8 characters from 4 character classes that enforce a significant password difference, a remembered history of 5 passwords, and a 90 day rotation policy, set the following Advanced Options (described in section 3):

```
Option: Security.PasswordQualityControl
Value: min=disabled,disabled,disabled,disabled,8 similar=deny

Option: Security.PasswordHistory
Value: 5

Option: Security.PasswordMaxDays
Value: 90
```

For more information, see the following topic:

4.4.3 Additional Authentication Mechanisms
Configuring SSH public key authentication is covered in section 4.2.8.1.

4.4.4 X.509 Certificate Validation and Authentication
ESXi offers methods to manage the certificate authorities trusted by the host. X.509 certificates presented by external services and by updates are verified against these certificate authorities. The CA roots used by the system for TLS connections are stored in the following location:

/etc/vmware/ssl/castore.pem

This location is not intended to be manipulated directly. See the VIM APIs and ESXCLI commands later in this section.

In the NIAP-approved configuration, the only secure outbound connection from an ESXi host is for remote syslog using TLS. (All other connections are inbound or secured using mechanisms other than TLS). See section 4.1.3 for information on how to configure CRL checking for remote syslog servers.

The CA roots used by the system for verifying software updates are stored in the following locations:

/usr/share/certs/vmware.cert
/usr/share/certs/vmpartner.cert

These locations are not modifiable.

ESXi offers the ability to locally store CRLs at the same time as CA roots are configured. This is intended for scenarios where the ESXi host lacks network connectivity to the CA root’s normal CRL distribution mechanism (for example, ESXi operating in an isolated network environment); this is an advanced configuration not evaluated for NIAP, as NIAP requirements presume accessibility of CRLs.

The VIM API HostCertificateManager (vim.host.CertificateManager) managed object offers methods to manage the certificate authorities trusted by the host. X.509 certificates presented by external services and by updates are verified against these certificate authorities.

<table>
<thead>
<tr>
<th>Method</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListCACertificateRevocationLists</td>
<td>Returns currently stored CRLs for CA roots used by this host.</td>
</tr>
<tr>
<td>ListCACertificates</td>
<td>Returns currently stored CA roots used by this host.</td>
</tr>
<tr>
<td>ReplaceCACertificatesAndCRLs</td>
<td>Installs new CA roots.</td>
</tr>
</tbody>
</table>

Table 6: VIM APIs for Certificate Management

For more information, see the following topic:

CA roots can also be configured using ESXCLI commands:

```
esxcli system security certificatestore add --file=<local-file>
esxcli system security certificatestore list
esxcli system security certificatestore remove --issuer=<issuer> --serial=<serial>
```

For more information, see the following topic:
ESXi’s CGI interface also offers a mechanism to directly set the `castore.pem` file using HTTPS put to the URI `/host/castore`. See section 5.5.1.

4.5 Management Functions (FMT)

4.5.1 Default Sharing Configurations
Virtual machines default to a configuration that does not contain any mechanisms for sharing data between virtual machines. This is not configurable (for example, it is not possible to change the default to include sharing). Administrators reconfiguring a VM (through the `vim.VirtualMachine.Reconfigure` VIM API) should use care to ensure any devices added to the virtual machine are configured correctly for sharing. Virtual network configuration is discussed further in section 4.3.1.

4.5.2 Timeouts
Shell timeouts for VIM, SSH, local (DCUI) shells, and Host Client are configurable through advanced options. See the following options in section 3:
- `Config.HostAgent.vmacore.soap.sessionTimeout`
- `UserVars.ESXiShellTimeOut`
- `UserVars.ESXiShellInteractiveTimeOut`
- `UserVars.DcuiTimeOut`
- `UserVars.HostClientSessionTimeout`

4.5.3 Management APIs (Consolidated)
ESXi operates in a model where all authenticated access is administrative. Non-administrative users (e.g. access to VMs) do not have any host configuration privileges at all. Non-administrative user virtual machine configuration privileges are only non-security-sensitive. (For example, a user can disconnect or “eject” a virtual CD-ROM image but cannot choose a different CD-ROM image to connect. Or a user can change screen resolution, which has no security impact).
The following table describes APIs used to perform certain management functions on an ESXi system. Some functions apply to the ESXi host and some apply to virtual machines.

<table>
<thead>
<tr>
<th>#</th>
<th>Function</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Installing ESXi updates</td>
<td>See section 4.6.1.</td>
</tr>
<tr>
<td>2</td>
<td>Configure password policy</td>
<td>See section 4.4.2.</td>
</tr>
<tr>
<td>3</td>
<td>Create, configure, and delete VMs</td>
<td><code>vim.Folder.CreateVM_Task</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>vim.VirtualMachine.ReconfigVM_Task</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>vim.ManagedEntity.Destroy_Task</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(vim.VirtualMachine base class)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>#createVm</td>
</tr>
<tr>
<td>4</td>
<td>Set default initial VM configurations</td>
<td>Default initial VM configurations are always “empty”. Override configuration can be supplied as the <code>config</code> parameter to CreateVM_Task. See: <a href="https://code.vmware.com/apis/358/vsphere/#/doc/vim.Folder.html#createVm">https://code.vmware.com/apis/358/vsphere/#/doc/vim.Folder.html#createVm</a> Override configuration can be supplied through the Create VM wizard in the ESXi Host Client: <a href="https://docs.vmware.com/en/VMware-vSphere/6.7/com.vmware.vsphere.html.hostclient.doc/GUID-FBEED81C-F9D9-4193-BDCC-CC4A60C20A4E.html">https://docs.vmware.com/en/VMware-vSphere/6.7/com.vmware.vsphere.html.hostclient.doc/GUID-FBEED81C-F9D9-4193-BDCC-CC4A60C20A4E.html</a></td>
</tr>
</tbody>
</table>

| 5 | Configure virtual networks | See section 4.3.1. |

| 6 | Configure and manage audit system and audit data | See section 4.1. |

| 7 | Configure VM access to physical devices | See section 4.3.2. |

| 8 | Configure inter-VM data sharing | See “Configure virtual networks” in this table. |

| 9 | Enable/disable hypercalls | See section 4.6.2. |

| 10 | Configure removable media policy | See section 4.6.3. |

| 11 | Configure cryptographic functionality | See section 4.2. |

| 12 | Change default authorization factors | See section 4.4. |

| 13 | Enable/disable screen lock | See section 4.5.2. |

| 14 | Configure screen lock inactivity timeout | See section 4.5.2. |

<p>| 15 | Configure remote connection inactivity timeout | See section 4.5.2. |</p>
<table>
<thead>
<tr>
<th></th>
<th>Configure lockout policy for unsuccessful authentication attempts</th>
<th>See section 4.4.1.</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Configure directory server</td>
<td>ESXi uses a local file for user directory services; no configuration is necessary. Use of an external directory server is not permitted in a NIAP configuration. <em>Note: ESXi does support using an Active Directory server for directory services. The security of this mechanism is not evaluated for NIAP and thus is not described here.</em></td>
</tr>
<tr>
<td>18</td>
<td>Configure audit/logging server</td>
<td>See section 4.1.3.</td>
</tr>
<tr>
<td>20</td>
<td>Configure banner</td>
<td>See section 4.7.1.</td>
</tr>
<tr>
<td>21</td>
<td>Connect/disconnect removable devices to/from a VM</td>
<td>See sections 4.3.2 and 4.6.3.</td>
</tr>
</tbody>
</table>
**Using ESXi Host Client:**

---

### 25  Suspend a VM

**vim.VirtualMachine.SuspendVM_Task**

See:

Using Host Client, click **Virtual Machines** in the inventory. Right-click a virtual machine and select **Power** and then **Suspend**.

---

### 26  Resume a VM

Resuming a VM is equivalent to starting a VM when the VM has a suspended state. See “Start a VM” in this table.

---

Table 7: Assorted Management Functions

---

### 4.6  Hypervisor Integrity (FPT)

#### 4.6.1  Trusted Updates

ESXi is made up of an “image profile” which describes a set of vSphere Installation Bundles (VIBs) that contain the actual software. A VIB is a signed ramdisk representing a component of the system, roughly analogous to an RPM or DEB on a Linux system. An “image profile” is a collection of VIBs; ESXi patches contain updated image profiles composed from a common set of VIBs.

ESXi updates are installed using ESXCLI, with commands in the `esxcli software` namespace.

The ESXi upgrade process is described in the **VMware ESXi Upgrade** documentation:

For NIAP configurations, use the ESXCLI instructions as described here:

ESXCLI may be used to list all installed VIBs and their current version, or the current image profile:
```
esxcli software vib list
esxcli software profile get
```

The general process is to put the ESXi host in maintenance mode, run an `esxcli software profile update` command (pointing to a URL or a .zip file transferred to the host through SSH), and reboot. Secure transfer of VIBs or the entire depot is not necessary, as the VIBs themselves are cryptographically signed by VMware and the update process will verify these signatures.

**Note:** Other upgrade mechanisms as described by the VMware ESXi Upgrade documentation might work correctly. However, these mechanisms require external services (for example, PXE...
booting), which are either not available in a NIAP configuration or where security cannot be evaluated.

The ESXi Host Client may be used to install individual VIBs, but not entire patch releases. Instructions are described here:

4.6.2 Hypercall Controls
Configurable hypercalls can be enabled or disabled by setting a particular “ExtraConfig” property for a virtual machine. This is done using the vim.VirtualMachine.ReconfigVM_Task API, passing a VirtualMachineConfigSpec with extraConfig containing a key and a value of “TRUE” or “FALSE” as described in the following topics:

These settings may be also configured using the ESXi Host Client:

The default settings for hypercalls are deemed secure; most hypercalls have no security impact and the remainder are secure by default. Disabling hypercalls might result in reduced virtual machine functionality; in some cases, it can render a virtual machine unable to boot (due to usage of hypercalls in BIOS/EFI). Modifying a hypercall configuration is not recommended. See section 5.6.2 for details about hypercalls.

4.6.3 Removable Devices and Media

4.6.3.1 USB devices
Removable USB devices (and any associated media) are covered in section 4.3.2.1.

4.6.3.2 Floppy devices
Floppy images in the form of files within a datastore can be connected to a virtual machine. This is most commonly used to install device drivers during guest operating system installation but may be used for other purposes. Floppy images support read and write access. VMware ESXi does not support a physical floppy device.

These configurations may be supplied through the VIM API by adding a vim.vm.device.VirtualFloppy to the virtual machine’s device list with the corresponding backing info.

<table>
<thead>
<tr>
<th>Type of Backing Object</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>vim.vm.device.VirtualFloppy.FileBackingInfo</td>
<td>Floppy image – a file attached to the virtual floppy device.</td>
</tr>
<tr>
<td>vim.vm.device.VirtualFloppy.DeviceBackingInfo</td>
<td>Not supported by VMware ESXi.</td>
</tr>
</tbody>
</table>
vim.vm.device.VirtualFloppy.RemoteDeviceBackingInfo | Floppy device forwarded over the network using the VM’s remote console. Remote client may use physical floppy or floppy image.

Table 8: BackingInfo Types for Virtual Floppy Devices  

These configurations may also be supplied via the ESXi Host Client.  

4.6.3.3 CD-ROM devices

CD-ROM images in the form of files within a datastore (ISO format) can be connected to a virtual machine. This is most commonly used to install a guest operating system. CD-ROM images support read access only.

A physical CD-ROM device can be connected to a virtual machine. This is rarely used in practice, as most servers running ESXi are physically isolated.

These configurations may be supplied through the VIM API by adding a vim.vm.device.VirtualCdrom to the virtual machine’s device list with the corresponding backing info.

<table>
<thead>
<tr>
<th>Type of Backing Object</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>vim.vm.device.VirtualCdrom.PassthroughBackingInfo</td>
<td>Physical CD-ROM device attached to the ESXi host.</td>
</tr>
<tr>
<td>vim.vm.device.VirtualCdrom.RemotePassthroughBackingInfo</td>
<td>CD-ROM device forwarded over the network using the VM’s remote console. Remote client may use physical CD-ROM or ISO image.</td>
</tr>
</tbody>
</table>

Table 9: BackingInfo Types for Virtual CD-ROM Devices  

These configurations may also be supplied via the ESXi Host Client.  
4.7 Accessing the Hypervisor (FTA)

4.7.1 Administrative Access Banner
ESXi has several configurable access “banners” before beginning an interactive administrative session. The contents are stored as strings, in Advanced Options on the system. The strings permit embedded newlines and support reasonable line lengths. When unset or set to an empty string, no message is displayed (unless noted as follows).

See the following options in section 3:
Annotations.WelcomeMessage
Config.Etc.Issue
Config.Etc.Motd
UserVars.HostClientWelcomeMessage

4.8 Secure Communication with the Hypervisor (FTP)

4.8.1 Isolating VM Networks from the Management Network
It is the responsibility of the administrator to provide isolation between VM networks and the management network.

Basic configuration of virtual switches, including the connection of virtual switches to the management network and VM networks, is covered in section 4.3.1.1.

A more detailed treatment of network configuration is available in the vSphere Security documentation:

4.8.2 Configuring Remote Audit Servers
Configuration of remote audit servers is covered in section 4.1.3.
5 DOCUMENTATION CLARIFICATIONS

The following information covers material not yet (or planned to be) present in public documentation.

5.1 Audit Configuration

Local audit log files are pre-allocated when configured. This prevents any error due to out-of-space during ordinary operation.

5.1.1 Audit Data and Auditable Events

This section describes the format of audit events in local and remote audit logs. Specific events are documented in Appendix A: Audit Events.

5.1.1.1 Audit Record Structure in Local Storage

An audit record is composed of several fields:
1) An ISO 8601 time stamp (the time when the audit record was generated).
2) An ASCII SPACE character.
3) The name of the program issuing the audit record.
4) An ASCII '[' character.
5) The process ID (PID) of the issuing program (in string form).
6) An ASCII ']' character.
7) An ASCII ':' character.
8) An ASCII SPACE character.
9) A structured data frame (which contains the audit record parameters).
10) An ASCII LF character (that is, a new line).

5.1.1.2 Audit Record Structure for remote syslog transmission

An audit record is composed of several fields:
1) The ASCII string <110>. This string indicates that this is an audit record issued at the LOG_INFO level.
2) An ISO 8601 time stamp (the time when the audit record was generated).
3) An ASCII SPACE character.
4) The identification string of the TOE (the "system name")
5) An ASCII SPACE character.
6) The name of the program issuing the audit record.
7) An ASCII '[' character.
8) The process ID (PID) of the issuing program (in string form).
9) An ASCII ']' character.
10) An ASCII ':' character.
11) An ASCII SPACE character.
12) A structured data frame (which contains the audit record parameters).
13) An ASCII LF character (that is, a new line).
5.1.1.3 Structured Data Frame Description

The language used for describing a structure data frame is taken from RFC 5424. A structured data frame is delimited by an open and a close square bracket ("[" and "]"). The format of the data within is:

[name@6876 paramName="paramValue" paramName="paramValue"
...
]

- RFC 5424 specifies that an SD-NAMESPACE must be a non-empty ASCII string which excludes whitespace, '=' and ']', and all control characters. It must be 32 characters or less in length.
- name@6876
  This is an RFC 5424 SD-ID (an SD-NAMESPACE). It consists of two parts, a name (which VMware is using for the audit event identifier (eventID)) which additionally cannot contain an at sign ('@'), and the IANA private enterprise number (VMware's is 6876) separated from the name by an at sign ('@').
- paramName
  This is an RFC 5424 PARAM-NAMESPACE (an SD-NAMESPACE). The same paramName can be specified for multiple paramName="paramValue" entries.
- paramValue
  This is an RFC 5424 PARAM-VALUE, a UTF-8 string (which can contain ASCII LF characters). If the string contains '"', '\', or ']', these characters must be escaped with a preceding backslash ('\'). A paramValue can be of any "reasonable" length (e.g. path names are OK). Note that a backslash followed by a character other than '"', '\', or ']', is not an error from the standpoint of validation: the '\' is dropped.

5.1.1.4 Structure Data Frame Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>comment</td>
<td>Optional comment.</td>
</tr>
<tr>
<td>deviceID</td>
<td>The deviceID associated with a USB device.</td>
</tr>
<tr>
<td>deviceLabel</td>
<td>The deviceLabel associated with a VM device.</td>
</tr>
<tr>
<td>HTTP.method</td>
<td>An HTTP method (name).</td>
</tr>
<tr>
<td>interface</td>
<td>Specifies &quot;console&quot; when appropriate.</td>
</tr>
<tr>
<td>ip</td>
<td>An IP address (V4 and V6 are permissible).</td>
</tr>
<tr>
<td>object</td>
<td>The object involved in the event (REQUIRED).</td>
</tr>
<tr>
<td>oldPath</td>
<td>The &quot;old&quot; path name.</td>
</tr>
<tr>
<td>operation</td>
<td>An operation identification string.</td>
</tr>
<tr>
<td>path</td>
<td>A path name.</td>
</tr>
<tr>
<td>pid</td>
<td>Product ID.</td>
</tr>
<tr>
<td>policy</td>
<td>The policy involved.</td>
</tr>
<tr>
<td>reason</td>
<td>Explanation, optional unless specifically noted otherwise.</td>
</tr>
<tr>
<td>result</td>
<td>&quot;success&quot; or &quot;failure&quot; (REQUIRED).</td>
</tr>
<tr>
<td>role</td>
<td>Required for connection events from outside the TOE.</td>
</tr>
<tr>
<td>status</td>
<td>Specifies &quot;connected&quot; or &quot;disconnected.&quot;</td>
</tr>
<tr>
<td>subject</td>
<td>Subject of the event. The user name or &quot;&quot; (ESXi) (REQUIRED).</td>
</tr>
</tbody>
</table>
5.1.1.5 Audit Record Examples

An audit record within local storage:
2018-10-09T20:13:10Z vmsyslogd[1000347563]: [audit.start@6876 subject="" object="" result="success"]

An audit record as transmitted off the TOE:
<110>2018-10-31T17:43:11Z esx-host.example.com esxcfg-syslog[1000353457]: [user@6876 subject="root" object="" result="success" comment="START EXPECT 7c12d6fa-defb-47b4-9384-7e83c5826121"]

5.2 Cryptographic Configuration

5.2.1 FIPS Certificates

ESXi makes use of the following CAVP certified Cryptographic Primitives.

<table>
<thead>
<tr>
<th>Cryptographic Module</th>
<th>Version</th>
<th>Security Policy</th>
<th>Algorithms (CAVP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vmkernel Cryptographic Module</td>
<td>1.0</td>
<td></td>
<td>AES, SHS, DRBG, HMAC (C 1172)</td>
</tr>
<tr>
<td>Vmkernel Cryptographic Module Loader</td>
<td>Not applicable</td>
<td></td>
<td>HMAC, SHA (C 1171)</td>
</tr>
<tr>
<td>Vmkernel DRBG Cryptographic Module</td>
<td>Not applicable</td>
<td></td>
<td>AES, DRBG (C 499)</td>
</tr>
<tr>
<td>Vmkernel DrbgLoader Cryptographic Module</td>
<td>Not applicable</td>
<td></td>
<td>HMAC, SHA256 (C 498)</td>
</tr>
<tr>
<td>VMware OpenSSL FIPS Object Module</td>
<td>2.0.20-vmw</td>
<td></td>
<td>DRBG, AES, SHS, HMAC, DSA, RSA, ECDSA, KAS-FPC, KAS-ECC (C 470)</td>
</tr>
</tbody>
</table>

Table 11: FIPS Certificates
5.2.2 Cryptographic Key Destruction

5.2.2.1 Volatile Key Destruction

By default, pages allocated for virtual machines, userspace applications, and kernel threads are zeroed out at allocation time. The hypervisor will always ensure that no non-zero pages are exposed to VMs or userspace applications. While this prevents exposing cryptographic keys from VMs or userworlds to other clients, these keys can stay present in memory for a long time if the memory isn't re-used.

A new configuration option MemEagerZero can be used to enable zeroing out userworld and guest memory pages (including VMM) when a userworld or guest exits. See section 4.2.3 for instructions to enable this option. By default, this option is disabled due to the performance impact of zeroing memory twice.

The MemEagerZero configuration option is a system wide setting that will impact all userworlds and guest pages. With the option disabled there is no specific time bound for when memory potentially containing cryptographic keys is zeroed; enabling this option will ensure cryptographic key destruction occurs immediately upon process exit. Cryptographic keys are considered in use until the process using the keys exists, due to data lifetime complications and potential use of language runtimes which make predicting key storage complicated.

For kernel threads, memory holding keys is zeroed out as soon as the secret is no longer needed. The rest of kernel memory will be automatically zeroed when the kernel terminates at power-off and reboot time.

5.2.2.2 Non-Volatile Key Destruction

ESXi has an unusual file system which operates as a ramdisk plus a backup mechanism that periodically writes a delta ("state.tgz") between the read-only software image to a FAT32 partition from which the system boots. All non-volatile system cryptographic keys would be stored only in this delta. When this backup mechanism executes, it overwrites all previous copies and intermediate files with zeroes, before unlinking the files and performing first an fsync then a sync operation to ensure these accesses reached disk.

Other non-volatile storage space is accessed as “datastores”. ESXi does not store system cryptographic keys on datastores, as datastores are for the storage of virtual machines and their metadata only.

5.2.2.3 Volatile Key Destruction in Vmkernel

Keys used in conjunction with vmkcrypt and vmkdrbg are either in use for the lifetime of the ESXi vmkernel (for example, to generate random numbers), or are used for a temporary purpose and are destroyed by vmkcrypt/vmkdrbg when that purpose is satisfied. All such keys are stored in volatile memory. All such keys are destroyed by vmkcrypt/vmkdrbg by zeroing memory when the resource is released.

The vmkdrbg module uses the same code as vmkcrypt, differing in symbol visibility only.

5.2.3 Entropy for Virtual Machines

ESXi does not implement any mechanism to supply entropy from the hypervisor to virtual machines. VMware has analyzed the problem and determined that entropy is a limited resource.
and supplying hypervisor entropy to virtual machines creates an opportunity for guests to induce denial-of-service attacks on the hypervisor. Instead, VMware recommends one of two approaches.

- **Guest-native entropy-gathering mechanisms.** Typical guests use interrupts (especially time-of-interrupt), storage block data, network block data, and HID (keyboard and mouse) input sources to generate entropy. These mechanisms work as well for virtual machines as they do for physical machines.  
  *As such, VMware believes a guest has sufficient entropy without depending on the hypervisor to supply additional entropy.*

- **Passthrough entropy-gathering mechanisms.** The simplest passthrough mechanisms are the RDRAND and RDSEED CPU instructions, which are executed directly on the CPU to supply randomness. ESXi does not modify the execution of RDRAND or RDSEED instructions. The NIAP evaluated configuration mandates a CPU with RDSEED support; VMware further recommends RDSEED support for any secure deployment. Usage of USB entropy-generating devices (and an associated in-guest USB driver) is also supported; see section 4.3.2.1 for USB passthrough configuration.

The following document provides a thorough discussion about the Linux kernel’s collection of entropy from multiple sources.


### 5.2.4 HTTPS protocol

ESXi implements the HTTPS protocol as specified in RFC 2818, using TLS as the transport.

### 5.2.5 SSH Protocol (Server)

ESXi implements the SSH protocol to comply with the following RFCs:

- [4251 SSH Protocol Architecture](https://www.ietf.org/rfc/rfc4251.txt)
- [4252 SSH Authentication Protocol](https://www.ietf.org/rfc/rfc4252.txt)
- [4253 SSH Transport Layer Protocol](https://www.ietf.org/rfc/rfc4253.txt)
- [4254 SSH Connection Protocol](https://www.ietf.org/rfc/rfc4254.txt)
- [5647 AES-GCM support for SSH](https://www.ietf.org/rfc/rfc5647.txt)
- [5656 EC Algorithms in SSH](https://www.ietf.org/rfc/rfc5656.txt)
- [6668 SHA-2 in SSH](https://www.ietf.org/rfc/rfc6668.txt)

### 5.2.6 SSH Protocol (Client)

ESXi contains an implementation of the SSH client protocol, used to make outbound connections from the ESXi host. This implementation is intended for troubleshooting use only and no evaluated claim is made about its security.
5.2.7 TLS Protocol
ESXi contains both server and client implementations of the TLS 1.2 protocol as specified in RFC 5246.
The software further contains implementations of TLS 1.1, TLS 1.0, and SSL 3 for compatibility with legacy environmental software. Usage of these protocols is off by default and is not permissible in a NIAP-validated configuration.
ESXi does not implement mutual TLS authentication (client certificates) as either a TLS client or TLS server. (NOTE: vSphere’s vCenter Server does use mutual TLS authentication, but is not within the TOE).

5.3 User Data Configuration

5.3.1 Memory Zeroization

5.3.1.1 Volatile Memory

Using VT-x with EPT, the virtual machine monitor can trap any initial access to volatile memory by a virtual machine. When this trap occurs, the virtual machine monitor obtains a zeroed page of memory from vmkernel, which it subsequently maps into the second-level page table, enabling the guest to access the page of memory.
Thus, any pages of memory are zeroed before becoming accessible to the guest.
For further details, see the whitepaper titled “Security of the VMware vSphere Hypervisor” at: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/whitepaper/techpaper/vmw-white-paper-secrty-vsphr-hyprvsr-uslet-101.pdf

5.3.1.2 Non-Volatile Memory

Virtual disks are implemented as VMDK files stored on a VMFS file system. The VMFS file system implements virtual disks in one of three modes: “lazy-zeroed thick,” “eager-zeroed thick,” and “thin.” All formats ensure data is zeroed before access by a virtual machine; the difference is the moment at which zeroing occurs. These formats are further described in the vSphere Storage documentation. https://docs.vmware.com/en/VMware-vSphere/6.7/com.vmware.vsphere.storage.doc/GUID-4C0F4D73-82F2-4B81-8AA7-1DD752A8A5AC.html

- The “eager-zeroed thick” format is a classic file, where the file contents are explicitly written with zeroes during the process of file creation, before the file is available for access.
- The “lazy-zeroed thick” format allocates but does not zero disk blocks when the file is created. VMFS (the file system) uses filesystem metadata to ensure that blocks are zeroed before use. This speeds disk creation but also ensures disk blocks are allocated to prevent out-of-space failure modes. Lazy zeroing is transparent and applied within the filesystem and is invisible to applications or virtual machines.
- The “thin” format is a sparse file which has metadata indicating a full size, but which does not allocate blocks to the disk until actual usage. This format can be created instantly but has a risk of out-of-space failures due to the ability to overcommit storage
space. Like “lazy-zeroed thick,” actual allocation and zeroing of disk blocks occurs within the filesystem before usage and is invisible to applications or virtual machines. Another mechanism exists: RDM (Raw Device Mapping) disks are passthrough to a storage volume (e.g. iSCSI or FibreChannel). In such cases, the storage volume itself is provided externally to ESXi, and memory zeroization effects would be a property of the (external) storage device or array. For further details, see the whitepaper titled “Security of the VMware vSphere Hypervisor” at: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/whitepaper/techpaper/vmw-white-paper-secrty-vsphr-hyprvsr-uslet-101.pdf

5.3.2 Hardware-based VM Isolation
VMware ESXi typically does not provide Guest VMs with direct access to any physical devices. This is accomplished using VT-x along with EPT to control which I/O ports and which memory the guest is able to directly access. VT-x allows all IN/OUT/INS/OUTS instructions to be trapped by the hypervisor. EPT allows ESX to control which pages are mapped into the guest's physical address space that limits which pages that guest can ever access. This enables only RAM pages to be mapped and to not include any physical MMIO addresses. VMware ESXi does, however, provide a PCI Passthrough mechanism to assign a specific PCI device to a virtual machine and allow the guest direct control over the device. When configured this way, ESX will map ranges of guest I/O ports to physical I/O ports on the device. The addresses of these ports might differ between the ports on the physical host and those in the virtual machine. As such ESX continues to use the VT intercepts provided by the CPU to intercept all I/O accesses. If the I/O is determined to be directed at one of the ports that is semantically mapped to the physical card, the VMM will proceed to perform the corresponding access using the correct physical I/O address on behalf of the guest. Additionally, ESX will map ranges of the guest's physical address space to MMIO ranges in the host physical address space that correspond to the PCI device that has been assigned to the virtual machine. Unlike with I/O ports, the CPU does provide a way to do this mapping directly using EPT. As such, ESX sets up the EPT tables to allow the guest direct access to the appropriate MMIO ranges using EPT.

5.3.3 Physical Resource Controls
VMware ESXi creates a distinct instance of the virtual machine monitor (VMM) and a corresponding userlevel process (VMX) for each running guest VM. A virtual machine is distinguished by its corresponding VMM or VMX instance, which are configured to access only permitted devices. The VMM uses Intel VT-X or AMD SVM to trap all guest access to physical hardware and either emulate the access in software, or (in the case of PCI Passthrough) forward the access to physical hardware. The NIAP evaluation does not include PCI Passthrough devices as no unused devices are available in the TOE configuration. PCI Passthrough is an uncommon configuration used for virtual machines which benefit from dedicated hardware. Thus, in the evaluated configuration all guest access to physical hardware are intercepted and emulated. When the VMM traps a guest access (IN or OUT instruction, or memory-mapped IO), the VMM determines how to emulate that access by looking up the specific IO port (IN/OUT instruction) or address (MMIO) and mapping the access to a specific passthrough device or specific virtual device and corresponding software implementation. In the PCI Passthrough scenario, this
mapping may forward directly to a physical device (see section 5.3.2). Any access to an IO port or MMIO address outside these configured mappings is “sunk” (ignored) on write, and reads all-ones (0xffffffff), which is the behavior specified for x86 CPUs when accessing non-existent ports or addresses.

Most virtual devices are implemented as a controller (a virtual PCI device on the guest’s PCI bus), with several devices accessed through the controller. Typical controllers are storage controllers (IDE, AHCI, NVMe, SCSI – including LsiLogic and PVSCSI variants) that would have virtual disks accessed through the controller, network interface controllers (vmxnet2, e1000, vmxnet3) that would be connected to various virtual switches in the host’s virtual network topology, USB controllers (called “USB Host Controllers”) that would have USB devices accessed through the controller, and floppy controllers that would be connected to floppy disk images, and several other controller types. In these cases, the virtual device controller is always a software emulation, and the virtual device controller examines and translates or emulates guest commands. For example, a virtual network controller will examine the Ethernet packet frame to check for validity and determine routing before passing the packet to the virtual networking layer on behalf of the guest, and a virtual storage controller will check for validity and translate a SCSI command into reads and writes to a particular virtual disk file on behalf of the guest.

The four devices that permit an administrator to associate a guest virtual machine with physical resources are PCI Passthrough, raw storage volumes, network connections and USB devices. For PCI Passthrough devices, the virtual machine configuration identifies a specific bus:slot:function (BDF) address on the host PCI bus, as described in section 4.3.2.4; the hypervisor will verify the corresponding PCI device is configured for access by this guest, and will ensure Intel VT-d or AMD IOMMU is used to ensure the PCI device can only access this guest. For network connections, the virtual machine configuration identifies a specific virtual portgroup (part of a virtual switch) in the host’s virtual network topology to which the virtual NIC will be connected, as described in section 4.3.1; the hypervisor will validate all packets before forwarding to the configured virtual portgroup. For USB devices, the virtual machine configuration identifies a specific USB device by vendor ID and product ID, two fields which are matched against an actual USB device, as described in section 4.3.2.1; the hypervisor will validate all USB packets before forwarding to a physical USB device. For raw storage volumes (RDM), the virtual machine configuration identifies a specific storage volume by device node name (for example, “/vmfs/devices/diskN” which must be an unmounted storage volume), as described in section 4.3.2.2; the hypervisor will validate any SCSI command before forwarding to the physical device.

Other virtual devices are implemented entirely in software, with no connection to a physical device. For example, keyboard and mouse are synthetic virtual devices which receive input from the guest virtual machine console connection. There is no mechanism for the host physical keyboard or mouse to provide input to a guest virtual machine. Likewise, the virtual graphics device implements a synthetic VGA, SVGA, and (optionally) 3D graphics accelerator. If available, the virtual graphics device will use a host graphics accelerator for acceleration purposes, by translating guest 3D commands into standard OpenGL commands and shaders and submitting these through a standard OpenGL software interface. There is no mechanism for the guest to directly control host graphics hardware.

As a general principle, a guest virtual machine has access to nothing by default, and individual virtual devices are added to the emulation. A guest virtual machine cannot access a device unless
it is explicitly added to the virtual machine’s configuration. This “whitelist” approach is desirable for virtualization as it better hides differences in physical hardware by avoiding the exposure of physical hardware to a guest.

5.4 Authentication Configuration

5.4.1 Login Mechanisms
ESXi implements several login mechanisms.

- **VIM API**: Use VIM’s `SessionManager.Login` method to supply a user name and password. This will set an HTTPS cookie for the HTTPS client containing a session ID, which may be used to authenticate future VIM API calls using distinct HTTPS connections during the same login session.
  The VIM API operates as SOAP method calls over HTTPS using the standard HTTPS port (443). Connections may be made based on hostname or IP address; ESXi’s self-signed TLS certificate specifies hostname only (Subject Alt Name DNS), but a customer-supplied TLS certificate may specify more.
- **Host Client (UI)**: Enabled by default. The Host Client’s web page accepts credentials (user name and password).
  Connections may be made based on hostname or IP address; ESXi’s self-signed TLS certificate specifies hostname only (Subject Alt Name DNS), but a customer-supplied TLS certificate may specify more.
- **ESXCLI**: The ESXCLI client accepts username and password parameters using the `--username` and `--password` flags.
  ESXCLI authenticates the server using TLS certificates when connecting. Clients should either install a corresponding CA certificate on the client, use the `--cacertsfile` flag to specify a CA certificate manually, or use the `--thumbprint` flag to specify a certificate thumbprint.
- **SSH**: Enable the SSH server (see section 4.2.8), then use an SSH client to connect to the host on the standard SSH port (22). The SSH server accepts user name and password authentication or public key authentication. Clients are responsible for verifying the ESXi public key.
- **CGI**: Specific HTTPS endpoints (see section 5.5.1) accept logins using HTTPS “Basic Auth” (username + password) for a single connection or using HTTPS cookie (VIM session ID) for multiple connections. Connections may be made based on hostname or IP address; ESXi’s self-signed TLS certificate specifies hostname only (Subject Alt Name DNS), but a customer-supplied TLS certificate may specify more.
- **DCUI**: Using the physical console, supply a user name and password to access a subset of management functions, intended to enable networking and SSH access for troubleshooting.

5.4.2 Password Complexity Enforcement
Password policy is handled by a common subsystem which uniformly implements password checking and, when changing passwords, enforces password complexity requirements. This subsystem is based on Pluggable Authentication Modules (PAM), a standard framework used by
many operating systems. All login mechanisms described above (section 5.4.1) verify passwords using the PAM subsystem.
The password policy may be changed only through the VIM API, including through the Host Client (UI) which is implemented using the VIM API. Password policy is not changeable using ESXCLI, SSH, CGI, or DCUI.

5.5 Management Functions

5.5.1 The CGI Interface
ESXi implements a small number of direct CGI commands and accessors, accessible over HTTPS only. These commands are intended for diagnostic purposes, for example, if the VIM API and ESXCLI are unavailable due to internal error. The accessors allow direct manipulation of particular files remotely.

<table>
<thead>
<tr>
<th>Command URL</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>/cgi-bin/vm-support.cgi</td>
<td>Collects a support dump (bundle of configuration files, log files, and core dumps). Fetches a .tgz file from the host.</td>
</tr>
<tr>
<td>/cgi-bin/esxcfg-info.cgi</td>
<td>Collects ESXi configuration info. Equivalent to /bin/esxcfg-info command. Text output.</td>
</tr>
<tr>
<td>/host/&lt;filename&gt;</td>
<td>Specific files can be edited using HTTPS GET and PUT commands. Selected security-relevant files:</td>
</tr>
<tr>
<td></td>
<td>• /host/ssl_cert – TLS public certificate</td>
</tr>
<tr>
<td></td>
<td>• /host/ssl_key – TLS private key (PUT only)</td>
</tr>
<tr>
<td></td>
<td>• /host/castore – TLS CA roots</td>
</tr>
<tr>
<td></td>
<td>• /host/ssh_root_authorized_keys – SSH public keys authorized for root login</td>
</tr>
<tr>
<td></td>
<td>• /host/ssh_host_rsa_key_pub – SSH public key</td>
</tr>
<tr>
<td></td>
<td>• /host/ssh_host_rsa_key – SSH private key (PUT only)</td>
</tr>
</tbody>
</table>

Table 12: CGI Interfaces

All CGI end points support two authentication mechanisms and require administrator-level authentication:

- HTTP Basic Auth. Allows user/password authentication for administrators. (This access is equivalent to full access through the VIM API).
- SOAP Cookie. A successful authentication using VIM API will set a session ID cookie. This cookie is normal for VIM API connections and is used to allow successive API calls to share the authenticated session of a prior call. The same cookie can also be supplied to a CGI handler, which will validate it internally using the VIM API.
5.6 Hypervisor Integrity

5.6.1 VMM Isolation
VMware ESXi does not provide a mechanism or ability for guest software to directly call platform APIs or to directly generate physical SMIs. Virtual SMIs are not correlated with physical SMIs.
Outside of the context of VMDirectPath I/O and support for nVidia GRID, no platform firmware, I/O ports, or MMIO registers are directly mapped into the address space or I/O space of the guest virtual machine.
While ESXi does contain a mechanism for updating microcode on the CPU, any attempt by guest software to update the microcode will be logged and then dropped.

5.6.2 Hypercalls
The VMware “hypercall” is known as the “Backdoor” and is implemented as a magic IN or OUT instruction where the CPU registers during the instruction carry special semantics. Appendix B (private documentation) describes all hypercalls.

5.6.3 Virtual Device Parameters
Most of VMware’s virtual devices are implemented as PCI devices that have no IO port exposure. Presence of these virtual devices can be detected through the PCI IDs described in the following table. All such virtual devices are “emulated” in that they do not connect directly to a physical device on the host.

<table>
<thead>
<tr>
<th>Vendor ID</th>
<th>Device ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1000 (LSI Logic)</td>
<td>0x0030</td>
<td>LSI Logic Controller (Parallel)</td>
</tr>
<tr>
<td>0x1000</td>
<td>0x0054</td>
<td>LSI Logic Controller (SaS)</td>
</tr>
<tr>
<td>0x15AD (VMware)</td>
<td>0x0405</td>
<td>VMware SVGA2 Graphics</td>
</tr>
<tr>
<td>0x15AD</td>
<td>0x0740</td>
<td>VMware VMCI</td>
</tr>
<tr>
<td>0x15AD</td>
<td>0x0750</td>
<td>Single port E1000</td>
</tr>
<tr>
<td>0x15AD</td>
<td>0x0760</td>
<td>Dual port E1000</td>
</tr>
<tr>
<td>0x15AD</td>
<td>0x0770</td>
<td>EHCI</td>
</tr>
<tr>
<td>0x15AD</td>
<td>0x0774</td>
<td>UHCI</td>
</tr>
<tr>
<td>0x15AD</td>
<td>0x0778</td>
<td>XHCI (Hardware versions 9 and lower)</td>
</tr>
<tr>
<td>0x15AD</td>
<td>0x0779</td>
<td>XHCI (Hardware versions 10 and newer)</td>
</tr>
<tr>
<td>0x15AD</td>
<td>0x07B0</td>
<td>VMware Vmxnet3</td>
</tr>
<tr>
<td>0x15AD</td>
<td>0x07C0</td>
<td>PVSCSI</td>
</tr>
<tr>
<td>0x15AD</td>
<td>0x07E0</td>
<td>AHCI</td>
</tr>
<tr>
<td>0x15AD</td>
<td>0x07F0</td>
<td>NVME</td>
</tr>
<tr>
<td>0x15AD</td>
<td>0x1975</td>
<td>HD Audio Codec</td>
</tr>
<tr>
<td>0x15AD</td>
<td>0x1977</td>
<td>HD Audio Controller</td>
</tr>
<tr>
<td>0x15AD</td>
<td>0x0820</td>
<td>RDMA</td>
</tr>
<tr>
<td>0x15AD</td>
<td>0x0830</td>
<td>vTPM</td>
</tr>
<tr>
<td>0x8086 (Intel)</td>
<td>0xB002</td>
<td>TXT Controller</td>
</tr>
<tr>
<td>0x8086</td>
<td>0x100f</td>
<td>E1000 single function</td>
</tr>
<tr>
<td>Device</td>
<td>Spec sheet</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Vmxnet3</td>
<td>Unavailable</td>
<td></td>
</tr>
<tr>
<td>VMCI</td>
<td>Unavailable</td>
<td></td>
</tr>
<tr>
<td>Super I/O</td>
<td>Implements standard Floppy, PS/2, Serial and Parallel ports. No specification available.</td>
<td></td>
</tr>
</tbody>
</table>

Table 13: PCI IDs

Most removable virtual devices solely implement memory mapped registers. A few virtual devices listed in the following table implement IO ports as well.

Table 14: IO Device Specification

Appendix C (private) describes IO ports for these virtual devices.

### 5.6.4 Virtualization Hardware Assists

Although VMware has a long history of extensively using binary translation, starting with version 6.7, ESX has eliminated the binary translator and always runs VMs using VT-x or AMD-V (depending on the brand of CPU used by the platform). VT-x and AMD-v are structurally and operationally similar. Though some implementation differences exist, they are not enough to need to discuss them separately. Both VT-x and AMD-v use a traditional trap and emulate model in which the guest code is run directly on the CPU in "guest" mode. When certain privileged operations occur, which require VMM involvement, the CPU generates an "exit" that performs a context switch to the hypervisor. The hypervisor then typically emulates the instruction that caused the exit and then returns the CPU to guest mode to resume executing guest code directly on the CPU. Additionally, some instructions that require special handling in a VM can be treated specially by the CPU, avoiding the need for an exit entirely to improve performance. This
support allows for a hypervisor to run an unmodified guest while providing the illusion that it runs directly on the CPU. Even though ESX no longer uses a binary translator, it does make use of a few techniques to improve performance that are reminiscent of binary translation. For instance, many exits occur repeatedly on the same instructions. ESX recognizes when this happens and optimizes its handling of those exits by constructing small code fragments that are customized for that guest instruction in context. Moreover, in some situations, guests perform multiple operations that cause exits in rapid succession. In those cases, ESX generates small code fragments that contain customized handlers for multiple guest instructions. These fragments never cover more than 16 guest instructions but are typically much smaller. This allows ESX to handle multiple instructions that would require an exit while only having to incur the hardware costs of a single exit.

5.6.4.1 Hardware-based Virtualization Assists on AMD CPUs

SVM (MD-V)
This extension provides the basic support for hardware assisted virtualization including introducing a guest mode and providing the ability to exit when executing sensitive instructions. VMware ESXi only supports SVM on K8L CPUs and later, which sets a minimum bar for the available features including support for RVI (hardware assisted two level paging).

TSC Rate
This extension provides the ability for a hypervisor to set a scale factor so that the apparent rate of the TSC value reported to the guest is different than the rate that the physical hardware runs at. This allows VMs to move from one machine to another and continue to observe a constant TSC rate. Without this feature the hypervisor must instead trap and emulate all RDTSC (and RDTSCP) instructions.

VMSAVE and VMLOAD Virtualization
This extension allows the processor to execute VMSAVE and VMLOAD instructions in guest mode without requiring a VMEXIT. Without this extension a hypervisor running a guest, which is itself using AMD-V, will have to trap and interpret these instructions.

5.6.4.2 Hardware-based Virtualization Assists on Intel CPUs

Intel VT-x (VMX)
As previously described, VT-x (that is, VMX) provides the basic support required for hardware-based virtualization including introducing a guest mode as well as providing support for exits when sensitive instructions are executed.

Unrestricted Guest
The original implementations of VT-x did not allow guest mode to operate in what is typically termed x86 real mode. This meant that the VMM would need to use interpretation or binary translation whenever the guest needed to use real mode such as when the virtual BIOS is executing. Unrestricted guest support extends the original VT-x implementation to allow real mode execution inside of guest mode eliminating the need for binary translation. VMware ESXi requires the presence of unrestricted guest mode.

NMI Window Exiting
On x86, NMI (non-maskable interrupts) are normally delivered right away. However, there are conditions in which NMIs are temporarily blocked by the CPU, such as following a MOVSS
instruction or after delivering an NMI but before an IRET is executed. NMI-window exiting allows the hypervisor to signal to the CPU that a virtual NMI is pending before resuming guest execution even if the guest is currently in a state where virtual NMIs are blocked. When the guest exits this state, the CPU generates an exit, allowing the hypervisor to deliver the NMI. Without NMI window exiting, the hypervisor is unable to resume guest execution if an NMI is pending and the guest has NMIs blocked. When this feature is unavailable, ESX is instead forced to interpret guest instructions until NMIs become unblocked.

**TSC Scaling**
This extension allows the hypervisor to specify a scale factor so that the apparent rate of the TSC counter as observed by the guest is different than the rate that the hardware TSC counter runs at. Scaling TSC values is important when a virtual machine moves between hosts of different TSC rates as it allows the VM to continue to perceive a single constant rate. On CPUs that lack TSC scaling assists, ESX instead traps and emulates all RDTSC (and RDTSCP) instructions.

**APIC Virtualization**
This extension allows the hypervisor to get special exit handling for memory access to the virtual APIC page. Without it ESX instead emulates MMIO accesses to the APIC using EPT violations.

**Virtualize X2APIC Mode**
This extension allows the hypervisor to avoid certain exits when the guest performs RDMSR or WRMSR instructions against MSRs that belong to the X2APIC. For certain MSR values in certain conditions, as documented in the Intel SDM section 29.5, the hardware can perform the correct virtualized behavior without hypervisor involvement. Without this extension the relevant RDMSR and WRMSR instructions are trapped by ESX and run through the interpreter.

**APIC-register Virtualization**
This extension allows the hypervisor to avoid certain exits when the guest performs MMIO operations to the virtual APIC's address range in memory. For certain regions on the page, in certain conditions the hardware can perform the correct virtualized behavior without hypervisor involvement. For detailed conditions see the Intel SDM chapter 29. Without this mode the relevant MMIO operations are trapped by ESX and interpreted.

**Virtual-interrupt Delivery**
If an interrupt needs to be delivered to a guest, but it cannot be delivered immediately (perhaps due to APIC priorities or holdoff), the hypervisor needs to postpone delivery. In this case, the hypervisor arranges for an exit to occur when conditions change that might allow the interrupt to be delivered. When the virtual-interrupt delivery extension is available, the hypervisor can instead pass the interrupt to hardware immediately and allow the CPU to monitor the relevant conditions and deliver the interrupt when possible without needing to exit to the hypervisor. ESX does not require this feature, but when present it can reduce the need for software interpretation of guest instructions.

**Process Posted Interrupts**
Normally, virtual interrupts delivered to the guest are synthetized by the hypervisor based on the emulation of virtual devices and do not necessarily correspond to a physical interrupt. When using PCI passthrough, however, physical interrupts might map to virtual interrupts in a 1:1 fashion. If this is true and if posted interrupts are supported by the hardware, this VT extension allows correctly tagged interrupts to be delivered directly to the guest without requiring an exit into the hypervisor. When this feature is unavailable ESX will instead proxy the interrupt to the guest using the interpreter if necessary.

**VMCS Shadowing**
This extension allows VMREAD and VMWRITE instructions operating in non-root mode to manipulate a shadow VMCS instead of causing exits. When unavailable, ESX will instead trap VMREAD and VMWRITE instructions and execute them through an interpreter.

PAUSE Loop Exiting
The PAUSE instruction is of interest to a hypervisor as an indication that the guest might be idle, and the hypervisor should consider scheduling a different execution context. Rescheduling only makes sense when the duration of the pause is relatively long. If the hypervisor requires the CPU to exit on every PAUSE instruction, a relatively cheap guest operation becomes extremely expensive. If the hypervisor chooses not to trap PAUSE, however, it might miss opportunities for rescheduling. The PAUSE-Loop Exiting extension indicates that the CPU can postpone any PAUSE exits until a time bound has been exceeded. If this feature is unavailable ESX might more often need to interpret PAUSE instructions.

Advanced EPT Exit Info
Although this relates to EPT, the availability of this feature allows ESX to ascertain more information about the reason for an EPT exit, which sometimes allows it to avoid invoking the interpreter. Without this feature ESX will more often need to interpret instructions when servicing an EPT violation. Its availability does not aid in avoiding software page tables.

5.6.4.3 Hardware-based Virtualization Memory-handling Assists

Extended Page Tables
This extension, which ESX requires, provides the basic support for two-level paging in the CPU. As described previously, this allows the hypervisor to avoid creating shadow page tables because the CPU will composite guest page tables with hypervisor provided second level page tables on the fly.

INVVPID Support (Base, Individual Address, Individual Context, All Context)
These EPT-related extensions indicate the availability for the CPU to respond to special forms of scoped TLB invalidation requests. ESX requires all four to be present to operate. The following table shows EPT-related extensions.

<table>
<thead>
<tr>
<th>Extension</th>
<th>Detailed Location</th>
<th>Description</th>
<th>Intel(R) Xeon(R) Gold 6126 CPU @ 2.60GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Required for power on</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Intel VT-x (aka VMX)</td>
<td>Bit 5 of ECX of leaf 1 of CPUID</td>
<td>Indicates the base support for Intel's VT-x virtualization extension.</td>
<td>X</td>
</tr>
<tr>
<td>Unrestricted guest</td>
<td>Bit 39 of IA32_VMX_PROCBASED_CLTS2</td>
<td>Indicates that the CPU can execute real mode code in non-root mode.</td>
<td>X</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NMI Window Exiting</td>
<td>Bit 54 of IA32_VMX_TRUE_PROCBASED_CLTS</td>
<td>Indicates that the CPU can track virtual NMI blocking windows.</td>
<td>X</td>
</tr>
<tr>
<td>Exit control with MSR</td>
<td>Bit 60 of IA32_VMX_TRUE_PROCBASED_CLTS</td>
<td>Allows the CPU to use MSR bitmaps to determine whether or not to exit.</td>
<td>X</td>
</tr>
<tr>
<td>TSC Scaling</td>
<td>Bit 57 of IA32_VMX_PROCBASED_CLTS2</td>
<td>CPU can scale the TSC values returned in non-root mode.</td>
<td>X</td>
</tr>
<tr>
<td>APIC Virtualization</td>
<td>Bit 32 of IA32_VMX_PROCBASED_CLTS2</td>
<td>CPU treats non-root mode writes to APIC addresses specially.</td>
<td>X</td>
</tr>
</tbody>
</table>
X2APIC Virtualization

Bit 36 of IA32_VMX_PROCBASED_CLTS2

CPU treats MSR reads and writes to XAPIC MSRs specially.

APIC Register Virtualization

Bit 40 of IA32_VMX_PROCBASED_CLTS2

CPU can virtualize access to certain APIC registers.

Virtual Interrupt Delivery

Bit 41 of IA32_VMX_PROCBASED_CLTS2

CPU can handle certain evaluation and delivery of pending interrupts.

Process Posted Interrupts

Bit 39 of IA32_VMX_TRUE_PINBASED_CLTS

Part of a mechanism to allow interrupts to be routed with less hypervisor involvement.

VMCS Shadowing

Bit 46 of IA32_VMX_PROCBASED_CLTS2

Non-root VMREAD and VMWRITE may access a shadow VMCS.

Pause Loop Exiting

Bit 42 of IA32_VMX_PROCBASED_CLTS2

Processor supports Pause Loop Exiting

Advanced EPT Exit Info

Bit 22 of IA32_VMX_EPT_VPID_CAP

Processor supports reporting advanced EPT exit info.

Required for power on

EPT

Bit 33 of IA32_VMX_PROCBASED_CLTS2

Processor supports extended page tables.

INVVPID support

Bit 32 of IA32_VMX_EPT_VPID_CAP

Processor supports INVVPID instruction.

INVVPID individual address support

Bit 40 of IA32_VMX_EPT_VPID_CAP

Processor supports individual address form of INVVPID.

INVVPID individual context support

Bit 41 of IA32_VMX_EPT_VPID_CAP

Processor supports individual context form of INVVPID.

INVVPID all context support

Bit 42 of IA32_VMX_EPT_VPID_CAP

Processor supports all context form of INVVPID.

AMD

Required for power on

SVM

Bit 2 of ECX of leaf 0x81 of CPUID

Indicates base support for AMD's SVM virtualization extension.

K8L or later

CPUID effective family indicates 0x10 or larger

Optional

TSC Rate

Bit 4 of EDX of leaf 0x8A of CPUID

CPU can scale the TSC values returned in guest mode.

VMSAVE and VMLOAD virtualization

Bit 15 of EDX of leaf 0x8A of CPUID

CPU can virtualize VMSAVE and VMLOAD instructions in guest mode

Table 15: CPU Feature Support in TOE's CPU

5.6.5 Execution Environment Mitigations

ESXi uses generally available execution environment mitigations. This document describes three broad areas, with different mitigations applicable to each area.

The most "exposed" security surface is userlevel binaries, which are the end point for both external communication (for example, HTTPS) and guest operations (for example, slow-path virtual machine emulation). These binaries are protected with ASLR, NX, and stack overflow protection.

The vmkernel is less exposed, but more sensitive. However, vmkernel is constrained by its runtime environment. Some mitigations require runtime or library support, which the vmkernel
lacks. Other mitigations require state to be stored where executable code cannot modify it, but a kernel can by definition modify all state. As a result, the vmkernel is protected with only (a) ASLR.

The monitor directly implements the "interface" the guest OS sees (for example, x86 instructions); as such, it is the most sensitive security surface. However, due to performance needs it generally cannot implement various security checks. Risk to the monitor is mitigated through non-environmental mitigations: the total size is kept small (about 100 KB program text), all code changes are carefully reviewed, and an extensive validation suite exhaustively tests all CPU instructions in both normal and error behaviors. This validation suite includes fuzz testing.

5.6.5.1 Address Layout Randomization

All userlevel executables are compiled as PIC (Position Independent Code) and linked as PIE (Position Independent Executable) to support relocation to different addresses. The vmkernel chooses random addresses when creating any program text segment, data segment, heap segment, thread stack, or other memory map (the "guest memory" map). The vmkernel is also loaded at a random address by the bootloader. In particular, the kernel's text and data segments are relocated to an address randomly chosen at each boot. Due to the need to load the kernel before initializing a secure RNG, implemented in the kernel, the random addresses used for vmkernel ASLR are not claimed to be cryptographically secure. The virtual machine monitor is loaded at a fixed address due to the need for a high-performance software page table walker (used to interpret the guest page table). The page table walker is among the hottest code paths and is designed such that compile-time-constant addresses are a necessary optimization.

5.6.5.2 Memory Execution Protection

All userlevel executables are loaded into and executed from page table mappings that implement W^X (write or execute but not both). This includes the use of -z noexecstack, so both text and stack segments are W^X. (Heap segments are inherently W^X at allocation time). The vmkernel behavior with respect to W^X is unspecified; this is not deemed a security concern as kernel page tables are no more difficult to modify than kernel text. The monitor does not implement W^X. See section 5.6.5 for an explanation of how the monitor is carefully controlled to not require this protection.

5.6.5.3 Stack Buffer Overflow Protection

All userlevel binaries are compiled using gcc with the -fstack-protector switch, which inserts stack smash protection for any buffers gcc heuristically determines to be at risk. (The -fstack-protector-all option is used during internal testing but comes at too much performance cost for regular use). The vmkernel lacks runtime support for stack overflow protection, due to not using standard compiler runtime support for process initialization. The standard compiler runtimes for stack overflow protection can only operate at userlevel. This is generally not a concern as the kernel also runs with small (16 KB) stacks, meaning kernel code already cannot store buffers on the stack for space reasons.
The monitor does not implement stack overflow protection. Like the kernel, it runs with small stacks which generally do not permit on-stack buffers.

5.6.5.4 Other Techniques

All userlevel binaries are compiled with _FORTIFY_SOURCE=1, which instruments many common libc functions (e.g. sprintf) for buffer overflow protections at runtime. The _FORTIFY_SOURCE set of behaviors are documented using glibc documentation. See 'man 7 feature_test_macros'.

All userlevel binaries are compiled with -z relro, which arranges for relocation tables (which must be written when a program loads) to be marked read-only after relocations are complete. Most binaries also use -z now to ensure this happens during load; the few exceptions are due to circular library dependencies which are incompatible with immediate loading. VMware confirms the presence of these options using a script that runs during the build and confirms all binaries packaged into the ESXi product meet these requirements. The build is instrumented to fail if any binary fails to show that all mitigations are enabled.

5.7 Accessing the Hypervisor

5.7.1 Administrative Access Banner

The access banner is displayed for all interactive sessions. Interactive sessions include:

- Host Client (UI)
- Local console access (DCUI)
- Remote SSH access

Usage of ESXi through direct API access is not considered an interactive session and thus does not display an access banner. The client of the API (which is external to ESXi) should have its own mechanism for displaying any necessary access banner.

In a typical vSphere deployment, ESXi is managed by vCenter Server (external to ESXi) and administrators would not have an interactive session directly to ESXi. This is similar to an Active Directory environment, where the interactive session is with the Domain Controller and the Domain Administrator would typically not have interactive sessions with individual workstations.

5.8 Secure Communication with the Hypervisor

5.8.1 The Management Network

VMware’s published best practices for vSphere deployments recommend isolating the management network (generally, vCenter Server and all ESXi hosts) from unnecessary access. This best practice coincides with the “Trustworthy Administrator” assumption in section 1.3. Common network isolation mechanisms are:

- **Physical.** Provisions the ESXi host with multiple network devices. Configure ESXi to treat one or several (for redundancy) NICs as the management network and configure a distinct set of NICs for attachment to virtual machines. The additional hardware and connection requirements make this a more resource-intensive configuration.
• **Logical (VLAN).** Allows ESXi to use the default NIC as a management network but configures virtual switches to operate over VLANs. In this way, virtual machine network traffic will be encapsulated and isolated from the management network. Configuring VLANs is beyond the scope of this document as configuration is also needed on devices external to the TOE.

• **None.** Virtual machines are connected to the same management network as the host. This is the default configuration; other modes require additional configuration.

A more detailed discussion of configuration of the management network is in the *vSphere Security* documentation:


A procedure for configuring virtual networking to place the management network and guest networking on physically isolated networks is:

1. Login to the VMware ESXi Host Client using an administrator account.
2. Click on the Networking entry in the Navigator pane.
3. Click on the “Virtual Switches” tab.
4. Click on “Add standard virtual switch”.
5. Enter a name of the new virtual switch (vSwitch1 in this example).
6. The new virtual switch (vSwitch1) will be assigned an unused physical NIC as the uplink adapter (vmnic1 in this example).
7. Select “Networking” in the left Navigator pane. Select the “Port groups” tab and click on the “Add port group” toolbar button.
8. Enter “Operational Network” for the name and select vSwitch1 for the “Virtual Switch.”
9. For all the virtual machines on this host, configure the VM’s network adapter so it is connected to “Operational Network”.
10. All VM traffic will now be physically separated from management traffic.

### 5.8.2 Secure Communication Using the Management Network

Management (administrative) operations occur exclusively using the VIM API and SSH. Limited management operations can occur using the local (physical) console through the DCUI, which is not accessible over the network.

- The VIM API is a SOAP protocol transmitted over HTTPS (default: port 443). This is the primary interface for ESXi and is secured by TLS. ESXi uses an X509 certificate to identify itself; clients are authenticated by the credentials they provide. The VIM API is not offered over non-TLS channels.

- The only outbound (ESXi-initiated) communication in the NIAP-evaluated configuration is to the remote syslog server. This communication occurs through TLS. The syslog server configuration includes “pinning” a particular X509 certificate to indicate the remote syslog server, which ESXi will verify when initiating communication.

- ESXi also offers an SSH interface (the default is disabled). This interface is intended for troubleshooting and recovery and might also be necessary for initial configuration in some scenarios.

- The DCUI is limited text-mode interface. It primarily exists to configure networking when setting up a new server; once networking is configured, the administrator would typically switch to the VIM API. In recovery and troubleshooting scenarios, the DCUI can also enable SSH.
The ESXi Host Client is a web application (a web page that runs within a client-side web browser) which uses JavaScript to make VIM API calls to ESXi over the management network. All ESXi Host Client functionality is implemented using the VIM API. A client connecting using the VIM API (or the Host Client UI) does so over the HTTP protocol (port 443). As part of using this protocol, the client is expected to verify the x509 certificate used by the ESXi host. Client verification is outside the scope of this document; typical approaches are Trust On First Use (TOFU), certificate thumbprints (verifying the certificate matches a known hash), and usage of a known Certificate Authority to sign ESXi’s x509 certificate. These approaches are generally implemented in the web browser (for Host Client UI) or command-line utilities (ESXCLI, for example). In common deployments, vCenter Server’s VMCA service functions as this trusted certificate authority, and the administrator can install the VMCA CA root in any client expected to communicate with ESXi. ESXi does expose limited functionality using unencrypted communications, as documented below. This functionality is not considered a security risk as access is read-only and the information disclosed is not sensitive. Example information is product version, and HTTP redirects to HTTPS services.

- HTTP (default: port 80). Contains HTTP redirects to HTTPS, and static XML or WSDL content describing host APIs. This content is a machine representation of public documentation.
- SLP (Service Locator Protocol) (default: port 427). Used for service discovery on a local network. Contains limited information about the host including a list of available services. Section 2.4.5 describes disabling this service.

### 5.8.3 User Interface Indicators

The only interface that supports direct viewing of a VM console is the Host Client (UI). Using that interface, administrators can open the VM console window as either an embedded window within the UI, as a distinct browser tab, as a distinct browser window, or through the VMware Remote Client (VMRC). Usage of VMRC is outside the scope of this evaluation. Within the Host Client (UI), VM consoles are identified by the name of the VM in the window’s title bar. The VM with input focus is the top-most window. The VIM API identifies virtual machines with a Managed Object Identifier (MOID), which is the unique identifier used within the API. The MOID is assigned automatically and uniquely within the TOE and may not be changed. As this identifier is meaningless to a user, the Host Client uses the human-readable “name” field within the Managed Object.