vRealize Suite Lifecycle Manager 1.x Security Hardening Guide

vRealize Suite 2017

vmware®

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

If you have comments about this documentation, submit your feedback to

docfeedback@vmware.com

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

vRealize Suite Lifecycle Manager 1.x Security Hardening Guide

The *vRealize Suite Lifecycle Manager 1.x Security Hardening Guide* helps users to evaluate and optimize the secure configuration of vRealize Suite Lifecycle Manager deployments.

The *vRealize Suite Lifecycle Manager 1.x Security Hardening Guide* describes verification and configuration of secure deployments for typical vRealize Suite Lifecycle Manager environments and provides information and procedures to help users make informed choices regarding security configuration.

Intended Audience

This information is intended for vRealize Suite Lifecycle Manager administrators and other users who are responsible for system security maintenance and configuration.

VMware Technical Publications Glossary

VMware Technical Publications provides a glossary of terms that might be unfamiliar to you. For definitions of terms as they are used in VMware technical documentation, go to http://www.vmware.com/support/pubs.

Hardening the VMware vSphere Environment

1

Assess the VMware vSphere® environment and verify that the appropriate level of vSphere hardening guidance is enforced and maintained.

As part of a comprehensively hardened environment, VMware vSphere infrastructure must meet security guidelines as defined by VMware.

For more guidance about hardening, see http://www.vmware.com/security/hardening-guides.html.

Securing the vRealize Suite Lifecycle Manager Appliance

2

Verify and update security settings for the vRealize Suite Lifecycle Manager appliance as necessary for your system configuration.

Configure security settings for your virtual appliances and their host operating systems. In addition, set or verify configuration of other related components and applications. In some cases, you need to verify existing settings, and in others you must change or add settings to achieve an appropriate configuration.

This section includes the following topics:

- vRealize Suite Lifecycle Manager Passwords
- Managing Secure Shell
- Harden Java Security

vRealize Suite Lifecycle Manager Passwords

You can change `root`, `sshuser`, `admin user` passwords using the common configuration setting in vRealize Suite Lifecycle Manager.

For information on changing vRealize Suite Lifecycle Manager passwords, see Change vRealize Suite Lifecycle Manager Passwords .

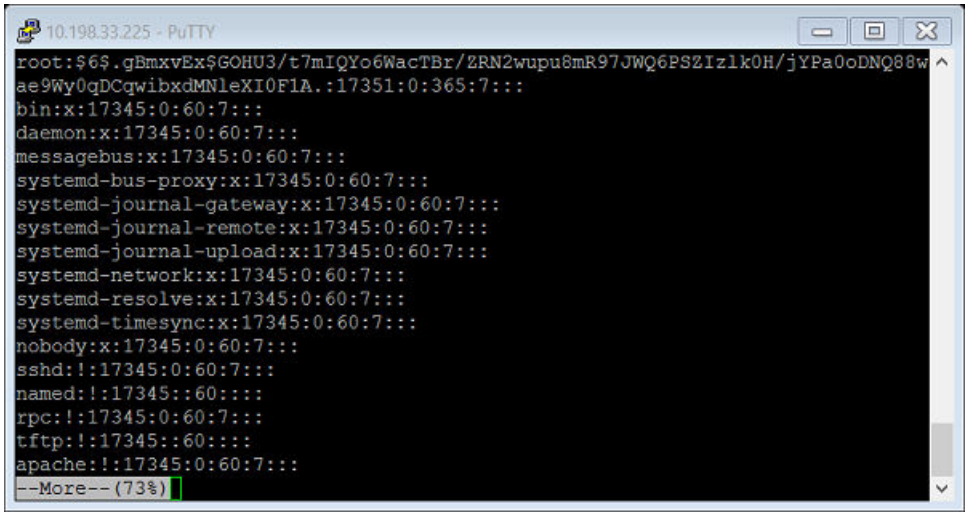
Verify Root Password Hash and Complexity

Verify that the root password meets your organization's corporate password complexity requirements.

Procedure

1 Log in as `root`, and run the `# more /etc/shadow` command.

The hash information is displayed.

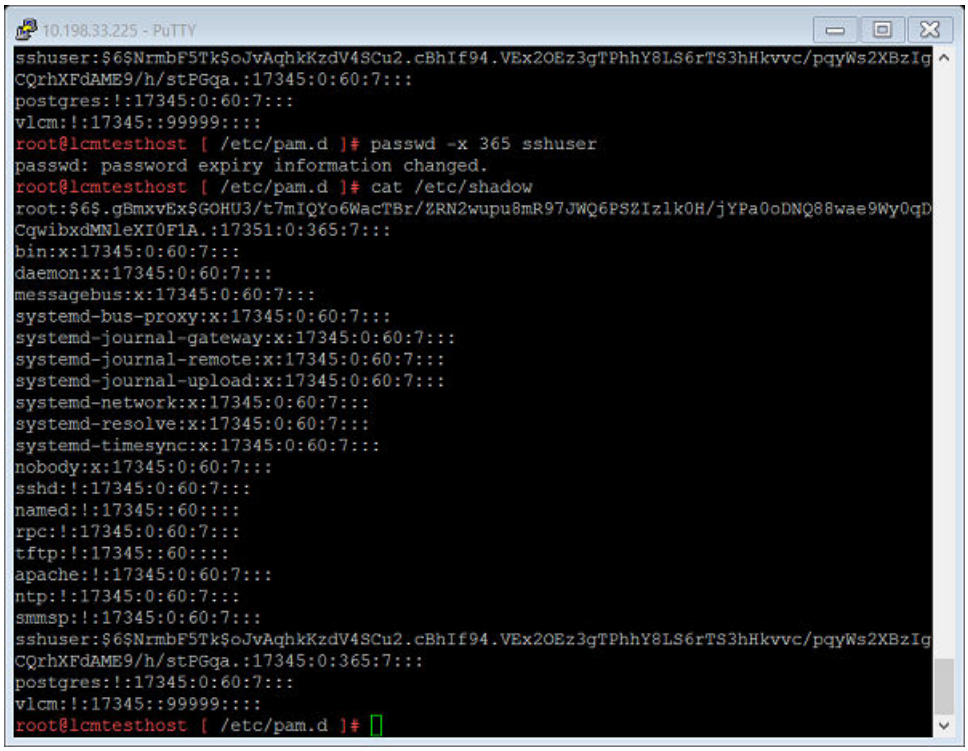


2 If the root password does not contain a `sha512` hash, run the `passwd` command to set a new password.

Managing Password Expiry

Configure all account password expirations in accordance with your organization's security policies.

By default, all hardened VMware virtual appliance accounts use a 60-day password expiration. On most hardened appliances, the root account is set to a 365-day password expiration. As a best practice, verify that the expiration on all accounts meets both security and operation requirements standards.



Managing Secure Shell

For remote connections, all hardened appliances include the Secure Shell (SSH) protocol. Use SSH only when necessary and manage it appropriately to preserve system security.

SSH is an interactive command-line environment that supports remote connections to VMware virtual appliances. By default, SSH access requires high-privileged user account credentials. Root user SSH activities generally bypass the role-based access control (RBAC) and audit controls of the virtual appliances.

Enable or Disable Secure Shell on vRealize Suite Lifecycle Manager

You can enable or disable SSH on vRealize Suite Lifecycle Manager as `root` user.

vRealize Suite Lifecycle Manager installs all products with SSH enabled by default. To disable SSH on products installed by vRealize Suite Lifecycle Manager, see that product's hardening guide.

For information on enabling and disabling Secure Shell by using the vRealize Suite Lifecycle Manager UI, see Enable or Disable SSH on vRealize Suite Lifecycle Manager.

Procedure

1. Log in as `root`.
2. (Optional) Run the command `# /usr/bin/systemctl {enable|disable} sshd` to enable SSH.
3. (Optional) Run the command `# /usr/bin/systemctl {start|stop} sshd` to disable SSH.

Harden the Secure Shell Server Configuration

Where possible, all VMware appliances have a default hardened configuration. Users can verify that their configuration is appropriately hardened by examining the server and client service settings in the global options section of the configuration file.

Procedure

- Open the `/etc/ssh/sshd_config` server configuration file on the VMware appliance, and verify that the settings are correct.

Setting	Setting Value in `ssh_config`
Client Protocol	`Protocol 2`
Client Gateway Ports	`GatewayPorts no`
GSSAPI Authentication	`GSSAPIAuthentication no`
CBC Ciphers	`Ciphers aes256-gcm@openssh.com,aes128-gcm@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr`
Message Authentication Codes	`MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha1-etm@openssh.com,hmac-sha2-512,hmac-sha2-256,hmac-sha1`

Example: Sample `ssh_config` File

```
#       $OpenBSD: ssh_config,v 1.30 2016/02/20 23:06:23 sobrado Exp $

# This is the ssh client system-wide configuration file.  See
# ssh_config(5) for more information.  This file provides defaults for
# users, and the values can be changed in per-user configuration files
# or on the command line.

# Configuration data is parsed as follows:
#  1. command line options
#  2. user-specific file
#  3. system-wide file
# Any configuration value is only changed the first time it is set.
# Thus, host-specific definitions should be at the beginning of the
# configuration file, and defaults at the end.

# Site-wide defaults for some commonly used options.  For a comprehensive
```

```
# list of available options, their meanings and defaults, please see the
# ssh_config(5) man page.

# Host *
#   ForwardAgent no
#   ForwardX11 no
#   RhostsRSAAuthentication no
#   RSAAuthentication yes
#   PasswordAuthentication yes
#   HostbasedAuthentication no
#   GSSAPIAuthentication no
#   GSSAPIDelegateCredentials no
#   BatchMode no
#   CheckHostIP yes
#   AddressFamily any
#   ConnectTimeout 0
#   StrictHostKeyChecking ask
#   IdentityFile ~/.ssh/identity
#   IdentityFile ~/.ssh/id_rsa
#   IdentityFile ~/.ssh/id_dsa
#   IdentityFile ~/.ssh/id_ecdsa
#   IdentityFile ~/.ssh/id_ed25519
#   Port 22
#   FipsMode no
#   Protocol 2
#   Cipher 3des
#   Ciphers aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-cbc,3des-cbc
#   MACs hmac-md5,hmac-sha1,umac-64@openssh.com,hmac-ripemd160
#   EscapeChar ~
#   Tunnel no
#   TunnelDevice any:any
#   PermitLocalCommand no
#   VisualHostKey no
#   ProxyCommand ssh -q -W %h:%p gateway.example.com
#   RekeyLimit 1G 1h
Ciphers aes256-gcm@openssh.com,aes128-gcm@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha1-etm@openssh.com,hmac-
sha2-512,hmac-sha2-256,hmac-sha1
GatewayPorts no
GSSAPIAuthentication no
```

Verify SSH Key File Permissions

To minimize the possibility of malicious attacks, maintain critical SSH key file permissions on your virtual appliance host machines.

Procedure

1 Check the SSH public key files by running the following command: `#ls -al /etc/ssh/*pub`

2 Verify that the files have the following permissions.

```
-rw-r--r-- 1 root root 609 date and time /etc/ssh/ssh_host_dsa_key.pub
-rw-r--r-- 1 root root 181 date and time /etc/ssh/ssh_host_ecdsa_key.pub
-rw-r--r-- 1 root root 101 date and time /etc/ssh/ssh_host_ed25519_key.pub
-rw-r--r-- 1 root root 401 date and time /etc/ssh/ssh_host_rsa_key.pub
```

3 Check the SSH private key files by running the following command: `ls –al /etc/ssh/*key`

4 Verify that the files have the following permissions.

```
-rw------- 1 root root  668 date and time /etc/ssh/ssh_host_dsa_key
-rw------- 1 root root  227 date and time /etc/ssh/ssh_host_ecdsa_key
-rw------- 1 root root  411 date and time /etc/ssh/ssh_host_ed25519_key
-rw------- 1 root root 1679 date and time /etc/ssh/ssh_host_rsa_key
```

5 Check the SSH configuration files by running the following command: `#lls –al /etc/ssh/*config`

6 Verify that the files have the following permissions.

```
-rw-r--r-- 1 root root 1914 date and time /etc/ssh/ssh_config
-rw------- 1 root root 3481 date and time /etc/ssh/sshd_config
```

Harden Java Security

Verify the settings in `/usr/java/jre-vmware/lib/security/java.security` meet the security requirement of your organization.

Procedure

- Open the `/usr/java/jre-vmware/lib/security/java.security` file on the VMware appliance, and verify that the settings are correct.

Setting	Setting Value in `java_security`
Secure Random Seed Source	`securerandom.source=file:/dev/random` `securerandom.strongAlgorithms=NativePRNGBlocking:SUN`
Default key and Trust Manager factory algorithm	`ssl.KeyManagerFactory.algorithm=SunX509` `ssl.TrustManagerFactory.algorithm=PKIX`
Enable OCSP	ocsp.enable=true
Algorithms disabled for certification path building and validation	`jdk.certpath.disabledAlgorithms=MD2, MD5, RSA keySize < 1024, \` `DSA keySize < 1024, EC keySize < 224`
Algorithms disabled for SSL/TLS processing	`jdk.tls.disabledAlgorithms= SSLv2, SSLv3, TLSv1, TLSv1.1, RC4, MD5withRSA, DH keySize < 768, EC keySize < 224`

Example: Sample `java_security` File

```
root@lcmtesthost [ /usr/java/jre-vmware/lib/security ]# cat java.security
#
# This is the "master security properties file".
#
# An alternate java.security properties file may be specified
# from the command line via the system property
#
#    -Djava.security.properties=<URL>
#
# This properties file appends to the master security properties file.
# If both properties files specify values for the same key, the value
# from the command-line properties file is selected, as it is the last
# one loaded.
#
# Also, if you specify
#
#    -Djava.security.properties==<URL> (2 equals),
#
# then that properties file completely overrides the master security
# properties file.
#
# To disable the ability to specify an additional properties file from
# the command line, set the key security.overridePropertiesFile
# to false in the master security properties file. It is set to true
# by default.

# In this file, various security properties are set for use by
# java.security classes. This is where users can statically register
# Cryptography Package Providers ("providers" for short). The term
# "provider" refers to a package or set of packages that supply a
# concrete implementation of a subset of the cryptography aspects of
# the Java Security API. A provider may, for example, implement one or
# more digital signature algorithms or message digest algorithms.
#
# Each provider must implement a subclass of the Provider class.
# To register a provider in this master security properties file,
# specify the Provider subclass name and priority in the format
#
#    security.provider.<n>=<className>
#
# This declares a provider, and specifies its preference
# order n. The preference order is the order in which providers are
# searched for requested algorithms (when no specific provider is
# requested). The order is 1-based; 1 is the most preferred, followed
# by 2, and so on.
#
# <className> must specify the subclass of the Provider class whose
# constructor sets the values of various properties that are required
# for the Java Security API to look up the algorithms or other
# facilities implemented by the provider.
#
```

```
# There must be at least one provider specification in java.security.
# There is a default provider that comes standard with the JDK. It
# is called the "SUN" provider, and its Provider subclass
# named Sun appears in the sun.security.provider package. Thus, the
# "SUN" provider is registered via the following:
#
#    security.provider.1=sun.security.provider.Sun
#
# (The number 1 is used for the default provider.)
#
# Note: Providers can be dynamically registered instead by calls to
# either the addProvider or insertProviderAt method in the Security
# class.

#
# List of providers and their preference orders (see above):
#
security.provider.1=sun.security.provider.Sun
security.provider.2=sun.security.rsa.SunRsaSign
security.provider.3=sun.security.ec.SunEC
security.provider.4=com.sun.net.ssl.internal.ssl.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider
security.provider.7=com.sun.security.sasl.Provider
security.provider.8=org.jcp.xml.dsig.internal.dom.XMLDSigRI
security.provider.9=sun.security.smartcardio.SunPCSC

#
# Sun Provider SecureRandom seed source.
#
# Select the primary source of seed data for the "SHA1PRNG" and
# "NativePRNG" SecureRandom implementations in the "Sun" provider.
# (Other SecureRandom implementations might also use this property.)
#
# On Unix-like systems (for example, Solaris/Linux/MacOS), the
# "NativePRNG" and "SHA1PRNG" implementations obtains seed data from
# special device files such as file:/dev/random.
#
# On Windows systems, specifying the URLs "file:/dev/random" or
# "file:/dev/urandom" will enable the native Microsoft CryptoAPI seeding
# mechanism for SHA1PRNG.
#
# By default, an attempt is made to use the entropy gathering device
# specified by the "securerandom.source" Security property.  If an
# exception occurs while accessing the specified URL:
#
#     SHA1PRNG:
#         the traditional system/thread activity algorithm will be used.
#
#     NativePRNG:
#         a default value of /dev/random will be used.  If neither
#         are available, the implementation will be disabled.
#         "file" is the only currently supported protocol type.
#
# The entropy gathering device can also be specified with the System
```

```
# property "java.security.egd". For example:
#
#   % java -Djava.security.egd=file:/dev/random MainClass
#
# Specifying this System property will override the
# "securerandom.source" Security property.
#
# In addition, if "file:/dev/random" or "file:/dev/urandom" is
# specified, the "NativePRNG" implementation will be more preferred than
# SHA1PRNG in the Sun provider.
#
securerandom.source=file:/dev/random

#
# A list of known strong SecureRandom implementations.
#
# To help guide applications in selecting a suitable strong
# java.security.SecureRandom implementation, Java distributions should
# indicate a list of known strong implementations using the property.
#
# This is a comma-separated list of algorithm and/or algorithm:provider
# entries.
#
securerandom.strongAlgorithms=NativePRNGBlocking:SUN

#
# Class to instantiate as the javax.security.auth.login.Configuration
# provider.
#
login.configuration.provider=sun.security.provider.ConfigFile

#
# Default login configuration file
#
#login.config.url.1=file:${user.home}/.java.login.config

#
# Class to instantiate as the system Policy. This is the name of the class
# that will be used as the Policy object.
#
policy.provider=sun.security.provider.PolicyFile

# The default is to have a single system-wide policy file,
# and a policy file in the user's home directory.
policy.url.1=file:${java.home}/lib/security/java.policy
policy.url.2=file:${user.home}/.java.policy

# whether or not we expand properties in the policy file
# if this is set to false, properties (${...}) will not be expanded in policy
# files.
policy.expandProperties=true

# whether or not we allow an extra policy to be passed on the command line
# with -Djava.security.policy=somefile. Comment out this line to disable
# this feature.
```

```
policy.allowSystemProperty=true

# whether or not we look into the IdentityScope for trusted Identities
# when encountering a 1.1 signed JAR file. If the identity is found
# and is trusted, we grant it AllPermission.
policy.ignoreIdentityScope=false

#
# Default keystore type.
#
keystore.type=jks

#
# Controls compatibility mode for the JKS keystore type.
#
# When set to 'true', the JKS keystore type supports loading
# keystore files in either JKS or PKCS12 format. When set to 'false'
# it supports loading only JKS keystore files.
#
keystore.type.compat=true

#
# List of comma-separated packages that start with or equal this string
# will cause a security exception to be thrown when
# passed to checkPackageAccess unless the
# corresponding RuntimePermission ("accessClassInPackage."+package) has
# been granted.
package.access=sun.,\
               com.sun.xml.internal.,\
               com.sun.imageio.,\
               com.sun.istack.internal.,\
               com.sun.jmx.,\
               com.sun.media.sound.,\
               com.sun.naming.internal.,\
               com.sun.proxy.,\
               com.sun.corba.se.,\
               com.sun.org.apache.bcel.internal.,\
               com.sun.org.apache.regexp.internal.,\
               com.sun.org.apache.xerces.internal.,\
               com.sun.org.apache.xpath.internal.,\
               com.sun.org.apache.xalan.internal.extensions.,\
               com.sun.org.apache.xalan.internal.lib.,\
               com.sun.org.apache.xalan.internal.res.,\
               com.sun.org.apache.xalan.internal.templates.,\
               com.sun.org.apache.xalan.internal.utils.,\
               com.sun.org.apache.xalan.internal.xslt.,\
               com.sun.org.apache.xalan.internal.xsltc.cmdline.,\
               com.sun.org.apache.xalan.internal.xsltc.compiler.,\
               com.sun.org.apache.xalan.internal.xsltc.trax.,\
               com.sun.org.apache.xalan.internal.xsltc.util.,\
               com.sun.org.apache.xml.internal.res.,\
               com.sun.org.apache.xml.internal.security.,\
               com.sun.org.apache.xml.internal.serializer.utils.,\
               com.sun.org.apache.xml.internal.utils.,\
               com.sun.org.glassfish.,\
```

```
                com.oracle.xmlns.internal.,\
                com.oracle.webservices.internal.,\
                oracle.jrockit.jfr.,\
                org.jcp.xml.dsig.internal.,\
                jdk.internal.,\
                jdk.nashorn.internal.,\
                jdk.nashorn.tools.,\
                com.sun.activation.registries.,\
                com.sun.browser.,\
                com.sun.glass.,\
                com.sun.javafx.,\
                com.sun.media.,\
                com.sun.openpisces.,\
                com.sun.prism.,\
                com.sun.scenario.,\
                com.sun.t2k.,\
                com.sun.pisces.,\
                com.sun.webkit.,\
                jdk.management.resource.internal.

#
# List of comma-separated packages that start with or equal this string
# will cause a security exception to be thrown when
# passed to checkPackageDefinition unless the
# corresponding RuntimePermission ("defineClassInPackage."+package) has
# been granted.
#
# by default, none of the class loaders supplied with the JDK call
# checkPackageDefinition.
#
package.definition=sun.,\
                   com.sun.xml.internal.,\
                   com.sun.imageio.,\
                   com.sun.istack.internal.,\
                   com.sun.jmx.,\
                   com.sun.media.sound.,\
                   com.sun.naming.internal.,\
                   com.sun.proxy.,\
                   com.sun.corba.se.,\
                   com.sun.org.apache.bcel.internal.,\
                   com.sun.org.apache.regexp.internal.,\
                   com.sun.org.apache.xerces.internal.,\
                   com.sun.org.apache.xpath.internal.,\
                   com.sun.org.apache.xalan.internal.extensions.,\
                   com.sun.org.apache.xalan.internal.lib.,\
                   com.sun.org.apache.xalan.internal.res.,\
                   com.sun.org.apache.xalan.internal.templates.,\
                   com.sun.org.apache.xalan.internal.utils.,\
                   com.sun.org.apache.xalan.internal.xslt.,\
                   com.sun.org.apache.xalan.internal.xsltc.cmdline.,\
                   com.sun.org.apache.xalan.internal.xsltc.compiler.,\
                   com.sun.org.apache.xalan.internal.xsltc.trax.,\
                   com.sun.org.apache.xalan.internal.xsltc.util.,\
                   com.sun.org.apache.xml.internal.res.,\
                   com.sun.org.apache.xml.internal.security.,\
```

```
                    com.sun.org.apache.xml.internal.serializer.utils.,\
                    com.sun.org.apache.xml.internal.utils.,\
                    com.sun.org.glassfish.,\
                    com.oracle.xmlns.internal.,\
                    com.oracle.webservices.internal.,\
                    oracle.jrockit.jfr.,\
                    org.jcp.xml.dsig.internal.,\
                    jdk.internal.,\
                    jdk.nashorn.internal.,\
                    jdk.nashorn.tools.,\
                    com.sun.activation.registries.,\
                    com.sun.browser.,\
                    com.sun.glass.,\
                    com.sun.javafx.,\
                    com.sun.media.,\
                    com.sun.openpisces.,\
                    com.sun.prism.,\
                    com.sun.scenario.,\
                    com.sun.t2k.,\
                    com.sun.pisces.,\
                    com.sun.webkit.,\
                    jdk.management.resource.internal.

#
# Determines whether this properties file can be appended to
# or overridden on the command line via -Djava.security.properties
#
security.overridePropertiesFile=true

#
# Determines the default key and trust manager factory algorithms for
# the javax.net.ssl package.
#
ssl.KeyManagerFactory.algorithm=SunX509
ssl.TrustManagerFactory.algorithm=PKIX

#
# The Java-level namelookup cache policy for successful lookups:
#
# any negative value: caching forever
# any positive value: the number of seconds to cache an address for
# zero: do not cache
#
# default value is forever (FOREVER). For security reasons, this
# caching is made forever when a security manager is set. When a security
# manager is not set, the default behavior in this implementation
# is to cache for 30 seconds.
#
# NOTE: setting this to anything other than the default value can have
#       serious security implications. Do not set it unless
#       you are sure you are not exposed to DNS spoofing attack.
#
#networkaddress.cache.ttl=-1

# The Java-level namelookup cache policy for failed lookups:
```

```
#
# any negative value: cache forever
# any positive value: the number of seconds to cache negative lookup results
# zero: do not cache
#
# In some Microsoft Windows networking environments that employ
# the WINS name service in addition to DNS, name service lookups
# that fail may take a noticeably long time to return (approx. 5 seconds).
# For this reason the default caching policy is to maintain these
# results for 10 seconds.
#
#
networkaddress.cache.negative.ttl=10

#
# Properties to configure OCSP for certificate revocation checking
#

# Enable OCSP
#
# By default, OCSP is not used for certificate revocation checking.
# This property enables the use of OCSP when set to the value "true".
#
# NOTE: SocketPermission is required to connect to an OCSP responder.
#
# Example,
#   ocsp.enable=true

#
# Location of the OCSP responder
#
# By default, the location of the OCSP responder is determined implicitly
# from the certificate being validated. This property explicitly specifies
# the location of the OCSP responder. The property is used when the
# Authority Information Access extension (defined in RFC 3280) is absent
# from the certificate or when it requires overriding.
#
# Example,
#   ocsp.responderURL=http://ocsp.example.net:80

#
# Subject name of the OCSP responder's certificate
#
# By default, the certificate of the OCSP responder is that of the issuer
# of the certificate being validated. This property identifies the certificate
# of the OCSP responder when the default does not apply. Its value is a string
# distinguished name (defined in RFC 2253) which identifies a certificate in
# the set of certificates supplied during cert path validation. In cases where
# the subject name alone is not sufficient to uniquely identify the certificate
# then both the "ocsp.responderCertIssuerName" and
# "ocsp.responderCertSerialNumber" properties must be used instead. When this
# property is set then those two properties are ignored.
#
# Example,
#   ocsp.responderCertSubjectName="CN=OCSP Responder, O=XYZ Corp"
```

```
#
# Issuer name of the OCSP responder's certificate
#
# By default, the certificate of the OCSP responder is that of the issuer
# of the certificate being validated. This property identifies the certificate
# of the OCSP responder when the default does not apply. Its value is a string
# distinguished name (defined in RFC 2253) which identifies a certificate in
# the set of certificates supplied during cert path validation. When this
# property is set then the "ocsp.responderCertSerialNumber" property must also
# be set. When the "ocsp.responderCertSubjectName" property is set then this
# property is ignored.
#
# Example,
#   ocsp.responderCertIssuerName="CN=Enterprise CA, O=XYZ Corp"

#
# Serial number of the OCSP responder's certificate
#
# By default, the certificate of the OCSP responder is that of the issuer
# of the certificate being validated. This property identifies the certificate
# of the OCSP responder when the default does not apply. Its value is a string
# of hexadecimal digits (colon or space separators may be present) which
# identifies a certificate in the set of certificates supplied during cert path
# validation. When this property is set then the "ocsp.responderCertIssuerName"
# property must also be set. When the "ocsp.responderCertSubjectName" property
# is set then this property is ignored.
#
# Example,
#   ocsp.responderCertSerialNumber=2A:FF:00

#
# Policy for failed Kerberos KDC lookups:
#
# When a KDC is unavailable (network error, service failure, etc), it is
# put inside a blacklist and accessed less often for future requests. The
# value (case-insensitive) for this policy can be:
#
# tryLast
#    KDCs in the blacklist are always tried after those not on the list.
#
# tryLess[:max_retries,timeout]
#    KDCs in the blacklist are still tried by their order in the configuration,
#    but with smaller max_retries and timeout values. max_retries and timeout
#    are optional numerical parameters (default 1 and 5000, which means once
#    and 5 seconds). Please notes that if any of the values defined here is
#    more than what is defined in krb5.conf, it will be ignored.
#
# Whenever a KDC is detected as available, it is removed from the blacklist.
# The blacklist is reset when krb5.conf is reloaded. You can add
# refreshKrb5Config=true to a JAAS configuration file so that krb5.conf is
# reloaded whenever a JAAS authentication is attempted.
#
# Example,
#   krb5.kdc.bad.policy = tryLast
```

```
#   krb5.kdc.bad.policy = tryLess:2,2000
krb5.kdc.bad.policy = tryLast

# Algorithm restrictions for certification path (CertPath) processing
#
# In some environments, certain algorithms or key lengths may be undesirable
# for certification path building and validation.  For example, "MD2" is
# generally no longer considered to be a secure hash algorithm.  This section
# describes the mechanism for disabling algorithms based on algorithm name
# and/or key length.  This includes algorithms used in certificates, as well
# as revocation information such as CRLs and signed OCSP Responses.
#
# The syntax of the disabled algorithm string is described as this Java
# BNF-style:
#   DisabledAlgorithms:
#       " DisabledAlgorithm { , DisabledAlgorithm } "
#
#   DisabledAlgorithm:
#       AlgorithmName [Constraint] { '&' Constraint }
#
#   AlgorithmName:
#       (see below)
#
#   Constraint:
#       KeySizeConstraint, CertConstraint
#
#   KeySizeConstraint:
#       keySize Operator DecimalInteger
#
#   Operator:
#       <= | < | == | != | >= | >
#
#   DecimalInteger:
#       DecimalDigits
#
#   DecimalDigits:
#       DecimalDigit {DecimalDigit}
#
#   DecimalDigit: one of
#       1 2 3 4 5 6 7 8 9 0
#
#   CertConstraint
#       jdkCA
#
# The "AlgorithmName" is the standard algorithm name of the disabled
# algorithm. See "Java Cryptography Architecture Standard Algorithm Name
# Documentation" for information about Standard Algorithm Names.  Matching
# is performed using a case-insensitive sub-element matching rule.  (For
# example, in "SHA1withECDSA" the sub-elements are "SHA1" for hashing and
# "ECDSA" for signatures.)  If the assertion "AlgorithmName" is a
# sub-element of the certificate algorithm name, the algorithm will be
# rejected during certification path building and validation.  For example,
# the assertion algorithm name "DSA" will disable all certificate algorithms
# that rely on DSA, such as NONEwithDSA, SHA1withDSA.  However, the assertion
# will not disable algorithms related to "ECDSA".
```

```
#
# A "Constraint" provides further guidance for the algorithm being specified.
# The "KeySizeConstraint" requires a key of a valid size range if the
# "AlgorithmName" is of a key algorithm.  The "DecimalInteger" indicates the
# key size specified in number of bits.  For example, "RSA keySize <= 1024"
# indicates that any RSA key with key size less than or equal to 1024 bits
# should be disabled, and "RSA keySize < 1024, RSA keySize > 2048" indicates
# that any RSA key with key size less than 1024 or greater than 2048 should
# be disabled. Note that the "KeySizeConstraint" only makes sense to key
# algorithms.
#
# "CertConstraint" specifies additional constraints for
# certificates that contain algorithms that are restricted:
#
#   "jdkCA" prohibits the specified algorithm only if the algorithm is used
#     in a certificate chain that terminates at a marked trust anchor in the
#     lib/security/cacerts keystore.  All other chains are not affected.
#     If the jdkCA constraint is not set, then all chains using the
#     specified algorithm are restricted.  jdkCA may only be used once in
#     a DisabledAlgorithm expression.
#     Example:  To apply this constraint to SHA-1 certificates, include
#     the following:  "SHA1 jdkCA"
#
# When an algorithm must satisfy more than one constraint, it must be
# delimited by an ampersand '&'.  For example, to restrict certificates in a
# chain that terminate at a distribution provided trust anchor and contain
# RSA keys that are less than or equal to 1024 bits, add the following
# constraint:  "RSA keySize <= 1024 & jdkCA".
#
# All DisabledAlgorithms expressions are processed in the order defined in the
# property.  This requires lower keysize constraints to be specified
# before larger keysize constraints of the same algorithm.  For example:
# "RSA keySize < 1024 & jdkCA, RSA keySize < 2048".
#
# Note: This property is currently used by Oracle's PKIX implementation. It
# is not guaranteed to be examined and used by other implementations.
#
# Example:
#   jdk.certpath.disabledAlgorithms=MD2, DSA, RSA keySize < 2048
#
#
jdk.certpath.disabledAlgorithms=MD2, MD5, RSA keySize < 1024, \
    DSA keySize < 1024, EC keySize < 224

# Algorithm restrictions for signed JAR files
#
# In some environments, certain algorithms or key lengths may be undesirable
# for signed JAR validation.  For example, "MD2" is generally no longer
# considered to be a secure hash algorithm.  This section describes the
# mechanism for disabling algorithms based on algorithm name and/or key length.
# JARs signed with any of the disabled algorithms or key sizes will be treated
# as unsigned.
#
# The syntax of the disabled algorithm string is described as follows:
#   DisabledAlgorithms:
```

```
#       " DisabledAlgorithm { , DisabledAlgorithm } "
#
#   DisabledAlgorithm:
#       AlgorithmName [Constraint]
#
#   AlgorithmName:
#       (see below)
#
#   Constraint:
#       KeySizeConstraint
#
#   KeySizeConstraint:
#       keySize Operator KeyLength
#
#   Operator:
#       <= | < | == | != | >= | >
#
#   KeyLength:
#       Integer value of the algorithm's key length in bits
#
# Note: This property is currently used by the JDK Reference
# implementation. It is not guaranteed to be examined and used by other
# implementations.
#
jdk.jar.disabledAlgorithms=MD2, MD5, RSA keySize < 1024

# Algorithm restrictions for Secure Socket Layer/Transport Layer Security
# (SSL/TLS) processing
#
# In some environments, certain algorithms or key lengths may be undesirable
# when using SSL/TLS.  This section describes the mechanism for disabling
# algorithms during SSL/TLS security parameters negotiation, including
# protocol version negotiation, cipher suites selection, peer authentication
# and key exchange mechanisms.
#
# Disabled algorithms will not be negotiated for SSL/TLS connections, even
# if they are enabled explicitly in an application.
#
# For PKI-based peer authentication and key exchange mechanisms, this list
# of disabled algorithms will also be checked during certification path
# building and validation, including algorithms used in certificates, as
# well as revocation information such as CRLs and signed OCSP Responses.
# This is in addition to the jdk.certpath.disabledAlgorithms property above.
#
# See the specification of "jdk.certpath.disabledAlgorithms" for the
# syntax of the disabled algorithm string.
#
# Note: This property is currently used by the JDK Reference implementation.
# It is not guaranteed to be examined and used by other implementations.
#
# Example:
#   jdk.tls.disabledAlgorithms=MD5, SSLv3, DSA, RSA keySize < 2048
jdk.tls.disabledAlgorithms=SSLv3, RC4, MD5withRSA, DH keySize < 768, \
    EC keySize < 224
```

```
# Legacy algorithms for Secure Socket Layer/Transport Layer Security (SSL/TLS)
# processing in JSSE implementation.
#
# In some environments, a certain algorithm may be undesirable but it
# cannot be disabled because of its use in legacy applications.  Legacy
# algorithms may still be supported, but applications should not use them
# as the security strength of legacy algorithms are usually not strong enough
# in practice.
#
# During SSL/TLS security parameters negotiation, legacy algorithms will
# not be negotiated unless there are no other candidates.
#
# The syntax of the legacy algorithms string is described as this Java
# BNF-style:
#   LegacyAlgorithms:
#       " LegacyAlgorithm { , LegacyAlgorithm } "
#
#   LegacyAlgorithm:
#       AlgorithmName (standard JSSE algorithm name)
#
# See the specification of security property "jdk.certpath.disabledAlgorithms"
# for the syntax and description of the "AlgorithmName" notation.
#
# Per SSL/TLS specifications, cipher suites have the form:
#       SSL_KeyExchangeAlg_WITH_CipherAlg_MacAlg
# or
#       TLS_KeyExchangeAlg_WITH_CipherAlg_MacAlg
#
# For example, the cipher suite TLS_RSA_WITH_AES_128_CBC_SHA uses RSA as the
# key exchange algorithm, AES_128_CBC (128 bits AES cipher algorithm in CBC
# mode) as the cipher (encryption) algorithm, and SHA-1 as the message digest
# algorithm for HMAC.
#
# The LegacyAlgorithm can be one of the following standard algorithm names:
#     1. JSSE cipher suite name, e.g., TLS_RSA_WITH_AES_128_CBC_SHA
#     2. JSSE key exchange algorithm name, e.g., RSA
#     3. JSSE cipher (encryption) algorithm name, e.g., AES_128_CBC
#     4. JSSE message digest algorithm name, e.g., SHA
#
# See SSL/TLS specifications and "Java Cryptography Architecture Standard
# Algorithm Name Documentation" for information about the algorithm names.
#
# Note: This property is currently used by the JDK Reference implementation.
# It is not guaranteed to be examined and used by other implementations.
# There is no guarantee the property will continue to exist or be of the
# same syntax in future releases.
#
# Example:
#   jdk.tls.legacyAlgorithms=DH_anon, DES_CBC, SSL_RSA_WITH_RC4_128_MD5
#
jdk.tls.legacyAlgorithms= \
        K_NULL, C_NULL, M_NULL, \
        DHE_DSS_EXPORT, DHE_RSA_EXPORT, DH_anon_EXPORT, DH_DSS_EXPORT, \
        DH_RSA_EXPORT, RSA_EXPORT, \
        DH_anon, ECDH_anon, \
```

```
        RC4_128, RC4_40, DES_CBC, DES40_CBC, \
        3DES_EDE_CBC

# The pre-defined default finite field Diffie-Hellman ephemeral (DHE)
# parameters for Transport Layer Security (SSL/TLS/DTLS) processing.
#
# In traditional SSL/TLS/DTLS connections where finite field DHE parameters
# negotiation mechanism is not used, the server offers the client group
# parameters, base generator g and prime modulus p, for DHE key exchange.
# It is recommended to use dynamic group parameters.  This property defines
# a mechanism that allows you to specify custom group parameters.
#
# The syntax of this property string is described as this Java BNF-style:
#   DefaultDHEParameters:
#       DefinedDHEParameters { , DefinedDHEParameters }
#
#   DefinedDHEParameters:
#       "{" DHEPrimeModulus , DHEBaseGenerator "}"
#
#   DHEPrimeModulus:
#       HexadecimalDigits
#
#   DHEBaseGenerator:
#       HexadecimalDigits
#
#   HexadecimalDigits:
#       HexadecimalDigit { HexadecimalDigit }
#
#   HexadecimalDigit: one of
#       0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f
#
# Whitespace characters are ignored.
#
# The "DefinedDHEParameters" defines the custom group parameters, prime
# modulus p and base generator g, for a particular size of prime modulus p.
# The "DHEPrimeModulus" defines the hexadecimal prime modulus p, and the
# "DHEBaseGenerator" defines the hexadecimal base generator g of a group
# parameter.  It is recommended to use safe primes for the custom group
# parameters.
#
# If this property is not defined or the value is empty, the underlying JSSE
# provider's default group parameter is used for each connection.
#
# If the property value does not follow the grammar, or a particular group
# parameter is not valid, the connection will fall back and use the
# underlying JSSE provider's default group parameter.
#
# Note: This property is currently used by OpenJDK's JSSE implementation. It
# is not guaranteed to be examined and used by other implementations.
#
# Example:
#   jdk.tls.server.defaultDHEParameters=
#       { \
#       FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1 \
#       29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD \
```

```
#       EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245 \
#       E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED \
#       EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE65381 \
#       FFFFFFFF FFFFFFFF, 2}

#
# The policy for the XML Signature secure validation mode. The mode is
# enabled by setting the property "org.jcp.xml.dsig.secureValidation" to
# true with the javax.xml.crypto.XMLCryptoContext.setProperty() method,
# or by running the code with a SecurityManager.
#
#   Policy:
#       Constraint {"," Constraint }
#   Constraint:
#       AlgConstraint | MaxTransformsConstraint | MaxReferencesConstraint |
#       ReferenceUriSchemeConstraint | KeySizeConstraint | OtherConstraint
#   AlgConstraint
#       "disallowAlg" Uri
#   MaxTransformsConstraint:
#       "maxTransforms" Integer
#   MaxReferencesConstraint:
#       "maxReferences" Integer
#   ReferenceUriSchemeConstraint:
#       "disallowReferenceUriSchemes" String { String }
#   KeySizeConstraint:
#       "minKeySize" KeyAlg Integer
#   OtherConstraint:
#       "noDuplicateIds" | "noRetrievalMethodLoops"
#
# For AlgConstraint, Uri is the algorithm URI String that is not allowed.
# See the XML Signature Recommendation for more information on algorithm
# URI Identifiers. For KeySizeConstraint, KeyAlg is the standard algorithm
# name of the key type (ex: "RSA"). If the MaxTransformsConstraint,
# MaxReferencesConstraint or KeySizeConstraint (for the same key type) is
# specified more than once, only the last entry is enforced.
#
# Note: This property is currently used by the JDK Reference implementation. It
# is not guaranteed to be examined and used by other implementations.
#
jdk.xml.dsig.secureValidationPolicy=\
    disallowAlg http://www.w3.org/TR/1999/REC-xslt-19991116,\
    disallowAlg http://www.w3.org/2001/04/xmldsig-more#rsa-md5,\
    disallowAlg http://www.w3.org/2001/04/xmldsig-more#hmac-md5,\
    disallowAlg http://www.w3.org/2001/04/xmldsig-more#md5,\
    maxTransforms 5,\
    maxReferences 30,\
    disallowReferenceUriSchemes file http https,\
    minKeySize RSA 1024,\
    minKeySize DSA 1024,\
    noDuplicateIds,\
    noRetrievalMethodLoops

#
# Serialization process-wide filter
#
```

```
# A filter, if configured, is used by java.io.ObjectInputStream during
# deserialization to check the contents of the stream.
# A filter is configured as a sequence of patterns, each pattern is either
# matched against the name of a class in the stream or defines a limit.
# Patterns are separated by ";" (semicolon).
# Whitespace is significant and is considered part of the pattern.
#
# If a pattern includes a "=", it sets a limit.
# If a limit appears more than once the last value is used.
# Limits are checked before classes regardless of the order in the sequence of patterns.
# If any of the limits are exceeded, the filter status is REJECTED.
#
#   maxdepth=value - the maximum depth of a graph
#   maxrefs=value  - the maximum number of internal references
#   maxbytes=value - the maximum number of bytes in the input stream
#   maxarray=value - the maximum array length allowed
#
# Other patterns, from left to right, match the class or package name as
# returned from Class.getName.
# If the class is an array type, the class or package to be matched is the element type.
# Arrays of any number of dimensions are treated the same as the element type.
# For example, a pattern of "!example.Foo", rejects creation of any instance or
# array of example.Foo.
#
# If the pattern starts with "!", the status is REJECTED if the remaining pattern
#   is matched; otherwise the status is ALLOWED if the pattern matches.
# If the pattern ends with ".**" it matches any class in the package and all subpackages.
# If the pattern ends with ".*" it matches any class in the package.
# If the pattern ends with "*", it matches any class with the pattern as a prefix.
# If the pattern is equal to the class name, it matches.
# Otherwise, the status is UNDECIDED.
#
#jdk.serialFilter=pattern;pattern

#
# RMI Registry Serial Filter
#
# The filter pattern uses the same format as jdk.serialFilter.
# This filter can override the builtin filter if additional types need to be
# allowed or rejected from the RMI Registry.
#
#sun.rmi.registry.registryFilter=pattern;pattern

#
# RMI Distributed Garbage Collector (DGC) Serial Filter
#
# The filter pattern uses the same format as jdk.serialFilter.
# This filter can override the builtin filter if additional types need to be
# allowed or rejected from the RMI DGC.
#
# The builtin DGC filter can approximately be represented as the filter pattern:
#
#sun.rmi.transport.dgcFilter=\
#    java.rmi.server.ObjID;\
```

```
#    java.rmi.server.UID;\
#    java.rmi.dgc.VMID;\
#    java.rmi.dgc.Lease;\
#    maxdepth=5;maxarray=10000
```

Ports and Protocols

3

As a security best practice, configure ports and protocols for all vRealize Suite Lifecycle Manager appliances and components in accordance with VMware guidelines.

Required Ports and Protocols

vRealize Suite Lifecycle Manager requires the following ports to be open to be fully functional. Disable all unneeded ports and protocols.

Server	Ports
vRealize Suite Lifecycle Manager appliance	443
SSH and SFTP	22