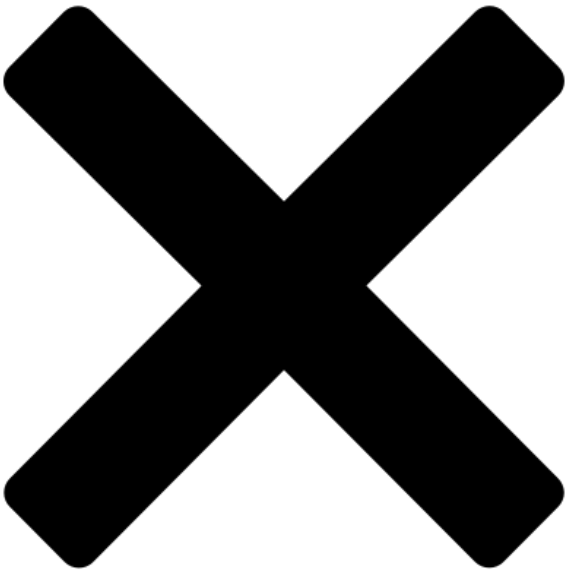
Detection, Containment, and Hardening Opportunities for Privileged Guest Operations, Anomalous Behavior, and VMCI Backdoors on Compromised VMware Hosts

mandiant.com/resources/blog/vmware-detection-containment-hardening



In Mandiant's initial publication of this vulnerability, we covered the attackers' exploitation of CVE-2023-20867, the harvesting of ESXi service account credentials on vCenter machines, and the implications of backdoor communications over VMCI socket. In this blog post, we will focus on the artifacts, logging options, and hardening steps to detect and prevent the following tactics and techniques seen being used by UNC3886:

- Both ESXi host and guest machine level logging options for Guest Operations
- vpxuser behavior indicative of anomalous usage
- Identifying open VMCI ports on ESXi hosts
- Multiple vCenter and ESXi containment and hardening recommendations to help deter future activity

For additional details, VMware has also released their guidance on the most recent CVE utilized by UNC3886.

Since the majority of threat actor operations cross the virtualization barrier between ESXi host and connected guest VMs, both successful and failed actions will have some sort of remnants available across both layers. The first section of this blog post will describe log options and configurations available on both ESXi hosts and guest VMs to identify when a threat actor performs successful guest operations.

ESXi Host and Guest Logging Options

The inherent behavior of Guest Operations can be partially tracked across Windows and Linux guest machines with proper EDR detections, but fall short in a few key situations. While EDR solutions can generally track file write interactions and new processes spawned from vmtoolsd.exe on Windows and from the vmtools daemon on Linux, they do not generally track file read operations. This particular interaction originally impacted Mandiant's ability to track UNC3886 transferring files from the guest VMs to the ESXi host. For this reason, Mandiant has identified and outlined additional logging sources and configurations that can be used to track when anomalous guest operations are occurring on both the ESXi host and guest machine level, regardless of the operation.

ESXi Host Logging

When Guest Operations are successfully performed on guest VMs from the ESXi Host, the ESXi host logs the operation action by default at the following path `/vmfs/volumes/…/<virtual machine hostname>/vmware.log`. Due to the location of this log, once a virtual machine is deleted from the ESXi host, the logs will be deleted alongside the `.vmdk` and `.vmx` files. This means by default only virtual machines currently present on the ESXi hosts can be reviewed for historical Guest Operations. Some examples of Guest Operations entries can be seen in Figure 1.

```
| vmx| I125: VigorTransportProcessClientPayload: opID=db88d87d seq=5552667: Receiving GuestOps.InitiateFileTransferFromGuest request.
| vcpu-0| I125: VigorTransport_ServerSendResponse opID=db88d87d seq=5552667: Completed GuestOps request.
| vmx| I125: VigorTransportProcessClientPayload: opID=db88d8cb seq=5552703: Receiving GuestOps.ListFiles request.
| vcpu-2| I125: VigorTransport_ServerSendResponse opID=db88d8cb seq=5552703: Completed GuestOps request.

| vmx| I125: VigorTransportProcessClientPayload: opID=dbb9bd43 seq=7241906: Receiving GuestOps.InitiateFileTransferToGuest request.
| vcpu-1| I125: VigorTransport_ServerSendResponse opID=dbb9bd43 seq=7241906: Completed GuestOps request.
| vmx| I125: VigorTransportProcessClientPayload: opID=dbb9cc78 seq=7244436: Receiving GuestOps.InitiateFileTransferFromGuest request.
| vcpu-0| I125: VigorTransport_ServerSendResponse opID=dbb9cc78 seq=7244436: Completed GuestOps request.
```

Figure 1: ESXi Host logging for Guest Operations

To focus on interactions between the ESXi Host and its guest machines in the vmware.log, filter for lines containing “GuestOps” (Here is a full list of Guest Operations). By default, these logs store up to six (6) separate log files, which rotate to a new file with every power on/off operation. The oldest log is removed in place of a new log once the max number of logs are reached. The following section further describes which configuration options can be used to customize the logs to the settings that best fit the environment.

Additional Host Level Logging Configurations

The following options can be used to configure ESXi host logging for Guest Operations to support additional retention, alternative rotation options, and location of the vmware.log output:

- **logging** – Set to either True or False, this value enables or disables Guest Operation logging for the virtual machine on the ESXi host
- **log.rotateSize** – By default, logs rotate to a new file when a power on or off operation occurs. This option changes the logs to only rotate to a new file once a specific size in bytes are met (Only available on ESXi 6.x+).
- **log.keepOld** – This value is set to six (6) by default and specifies the number of log rotations retained. Mandiant suggests increasing this value based on available storage and number of VMs on the ESXi host.
- **log.fileName** – Sets the filename used to save the Guest Operation logs. If a full path is provided, Guest Operations will be written to that directory.
- **vmx.log.destination** – When this value is set to syslog-and-disk, logs will be forwarded to both the ESXi local datastore and the syslog server defined in the `Host syslog settings`. The vmx.log.syslogID value must be set correctly to be able to identify the virtual machines associated with these logs.
- **vmx.log.syslogID** – A unique identifier to associate the logs forwarded to the syslog server with a guest machine. This is most commonly set to the name of the Virtual Name if a unique guest machine naming schema is being enforced.

Additional information surrounding these logging configurations can be found within the following VMware article.

In addition to ESXi host logging, Mandiant has identified optional logging on guest VMs with VMware Tools (vmtools) installed, which enable high-level, or detailed logging options, on Guest Operations being performed from the ESXi host.

Guest Machine Logging (vmsvc.log)

By default, the `vmsvc.log` is not enabled by vmtools on guest machines, but this log provides a few different levels of visibility into any Guest Operations being executed on the host when enabled. This visibility ranges from seeing the names of the operations being performed when message level logging is enabled, to full visibility into files being transferred and command line arguments being executed if debugging is enabled.

Message Level Logging (vmsvc.log)

Having the `vmsvc.log` enabled at the message level enables partial visibility into the types of actions being taken against the guest machines by the ESXi host due to VixTools operation codes being logged. The VIXTools API (Vix) is a library designed for writing scripts and programs to manipulate virtual machines and is the backend that drives Guest Operations. Since Guest Operations are directly related to the VixTools API, it is possible to translate Vix operational codes to the Guest Operations used to invoke them. Figure 2 shows an example line in the `vmsvc.log` which contains these Vix codes (denominated as X) and can be translated into an equivalent Guest Operation:

Figure 2: Example VixTools Events in message level logging (vmsvc.log)

```
[message] [vix] VixTools_ProcessVixCommand: command X
```

To translate these Vix codes, the following source code contains a list of all possible Vix operational codes. Some of the key Vix codes which are related to Guest Operations seen being used by the attacker can be seen in Table 1:

Table 1: Vix Operational Codes to Guest Operations

VixCommand OpCode	Vix Command Name	Guest Operation Equivalent
177	VIX_COMMAND_LIST_FILES	ListFilesInGuest
185	VIX_COMMAND_START_PROGRAM	StartProgramInGuest
186	VIX_COMMAND_LIST_PROCESSES_EX	ListProcessesInGuest
188	VIX_COMMAND_INITIATE_FILE_TRANSFER_FROM_GUEST	InitiateFileTransferFromGuest
189	VIX_COMMAND_INITIATE_FILE_TRANSFER_TO_GUEST	InitiateFileTransferToGuest
84	VMXI_HGFS_SEND_PACKET_COMMAND	Accompanies 188 and 189 to send and receive files

Message level logging in the `vmsvc.log` can be a consistent tool to alert on anomalous Guest Operations occurring on guest machines to prompt further investigation. This level of logging still falls short in the same manner as many EDR solutions if files are being copied from the guest machine to the ESXi host as the specific files cannot be identified. This is where debugging level logging is useful.

Debug Level Logging (vmsvc.log)

For full visibility into the Guest Operations within the `vmsvc.log`, debugging level logging can be enabled. This allows visibility into:

- Events showing that impersonation of the root user is occurring
- The commands and arguments being passed on to the guest machine from the ESXi host
- The number of bytes sent to and from the ESXi host and guest machine during guest operations
- The source path, destination path, file names, and size of file being sent during file transfer operations

An example of debugging level logging for the “StartPrograminGuest” Guest Operation can be found in Figure 3 for Linux guest VMs and Figure 4 for Windows guest VMs.

Linux Guest Machine:

Figure 3: Linux guest VM Debug Level Logging (vmsvc.log)

```
[debug] [vmsvc] RpcIn: received 198 bytes, content:"Vix_1_Relayed_Command
"588cea63b2ecd6cd"\00\01\00\0d\d0\05\00\9d\00\00\003\00\00\00-
\00\00\00=\00\00\00\0d\b9\00\00\00\00\00\00..."

[message] [vix] VixTools_ProcessVixCommand: command 185

[debug] [vix] VixToolsImpersonateUser: successfully impersonated user _ROOT_

[debug] [vix] VixTools_StartProgram: User: _ROOT_ args: progamPath: '/bin/ls', arguments: '"-R"
"/var/log"', workingDir: ''

[debug] [vmsvc] Executing async command: '"/bin/ls" "-R" "/var/log"' in working dir '/root'

[debug] [vix] VixToolsStartProgramImpl: started '"/bin/ls" "-R" "/var/log"', pid 8817

[debug] [vix] VixTools_StartProgram: returning '8817'

[message] [vix] VixTools_StartProgram: opcode 185 returning 0

[debug] [vix] ToolsDaemonTcloReceiveVixCommand: command 185, additionalError = 0

[debug] [vmsvc] RpcIn: sending 1242 bytes
```

Windows Guest Machine:

Figure 4: Windows guest VM Debug Level Logging (vmsvc.log)

```
[   debug] [vmsvc] [3116] RpcIn: received 253 bytes, content:"Vix_1_Relayed_Command
"25642877bd645a32"\00\01\00\0dÐ\05\00Ô\00\00\003\00\00\00X\00\00\00I\00\00\00\0d¹\00\00\00\00\00\00\00ÿÿÿÿ..."

[ message] [vix] [3116] VixTools_ProcessVixCommand: command 185

[   debug] [vmsvc] [3116] VMTools_ConfigGetBoolean: Returning default value for '[guestoperations]
disabled'=FALSE (Not found err=3).

[   debug] [vmsvc] [3116] VMTools_ConfigGetBoolean: Returning default value for '[guestoperations]
StartProgramInGuest.disabled'=FALSE (Not found err=3).

[   debug] [vix] [3116] VixToolsImpersonateUser: successfully impersonated user _ROOT_

[   debug] [vix] [3116] VixTools_StartProgram: User: _ROOT_ args: progamPath: 'c:\windows\system32\cmd.exe',
arguments: '"/c dir /od /s /a c:\\ > C:\\Test.txt"', workingDir: ''

[   debug] [vmsvc] [3116] ProcMgr_ExecAsync: Executing async command: "c:\windows\system32\cmd.exe" "/c dir
/od /s /a c:\\ > C:\\Test.txt"

[   debug] [vmsvc] [3116] Spawned sub-process 888

[   debug] [vix] [3116] VixToolsStartProgramImpl: started '"c:\windows\system32\cmd.exe" "/c dir /od /s /a
c:\\ > C:\\Test.txt"', pid 4812

[   debug] [vix] [3116] VixToolsUnimpersonateUser: Faking unimpersonate

[   debug] [vix] [3116] VixTools_StartProgram: returning '4812'

[ message] [vix] [3116] VixTools_StartProgram: opcode 185 returning 0

[   debug] [vix] [3116] ToolsDaemonTcloReceiveVixCommand: command 185, additionalError = 0

[   debug] [vmsvc] [3116] RpcIn: sending 12 bytes
```

Both configurations can be set to `message` or `debug`, depending on the visibility and storage capabilities of the environment it's being enabled on.

Mandiant advises that a test environment is staged with vmsvc logging enabled at both levels to better identify the performance impact this may have on each individual environment, prior to additional logging levels being enabled in production.

Configuring Extended Logging on Windows and Linux Machines

To enable the `vmsvc.log` on guest VMs with vmtools installed, locate the `tools.conf` file which can be found at one of the locations specified in Figure 5.

Guest OS	Path [1]
Windows XP, Windows Server 2000, and Windows Server 2003	C:\Documents and Settings\All Users\Application Data\VMware\VMware Tools\tools.conf [2]
Windows Vista, Windows 7, and Windows Server 2008 [3]	C:\ProgramData\VMware\VMware Tools\tools.conf [4]
Linux, Solaris and FreeBSD	/etc/vmware-tools/tools.conf
FreeBSD with open-vm-tools 10.1.x or later	/usr/local/share/vmware-tools/tools.conf
Mac OS X	/Library/Application Support/VMware Tools/tools.conf

Figure 5: Location of vmsvc.log configuration file per operating system

Within `tools.conf`, insert the appropriate configuration based on the relevant operating system seen in both Figure 6 and Figure 7. The VMware Tools service will need to be restarted for the new logging level to take effect.

Figure 6: Windows tools.conf configurations (Use forward slashes)

```
[logging]

log = true
vmsvc.level = debug
vmsvc.handler = file
vmsvc.data = c:/Windows/Temp/vmsvc.log
```

Figure 7: Linux tools.conf configurations

```
[logging]
log = true

vmsvc.level = debug
vmsvc.handler = file
vmsvc.data = /tmp/vmsvc.log
```

For further instructions on how to apply these configurations within Windows and Linux guest machines, please follow VMware's guide.

To enable log forwarding to syslog or other sources such as the ESXi Host, please reference the "syslog" section of the default `tools.conf` configuration file.

The aforementioned logging options across ESXi host and guest VMs provide visibility into Guest Operations occurring between host and guest VM. However, they do not log the presence of VMCI backdoors being deployed or the communication that occurs when a threat actor is connecting to a VMCI backdoor. To account for this, the following section details how to identify if a binary has an open VMCI port on an ESXi host.

Detecting Open VMCI Sockets on ESXi Hosts

While VMCI provides threat actors with a flat network to listen on and connect over for extended persistence in virtualized environments, this same concept enables defenders to perform checks for VMCI backdoors with similar flexibility. Since VMCI sockets on ESXi hosts are open to all guest machines running underneath the ESXi host, any guest machine running underneath the host can sniff VMCI port traffic to identify anomalies.

While scanning for open VMCI ports sounded like a viable approach at first, it becomes unrealistic when attempting to account for the entire port range. As of this blog post, Mandiant has not observed threat actors utilizing port numbers outside of the traditional 65,535 range (the range for TCP/UDP ports). However, the port can range from 0-2,147,483,647 as the port number field for VMCI socket address is 32-bits. This makes scanning a time-consuming process that is not realistic in a production environment.

At the time of writing, the most consistent way Mandiant has identified VMCI ports open on ESXi hosts is through the usage of the `lsof -A` command. This command displays file information and metadata about files opened by processes, and in effect on ESXi hosts, it also lists processes that actively have open VMCI ports. As seen in Figure 8, when a process is listening on a VMCI port, `lsof` will show the process as containing a type `SOCKET_VMCI` and a description field with the value `{no file name}`. Mandiant has guidance on how to dump ESXi host process memory in the following blog post to further inspect the offending binary so the exact port number can be identified.

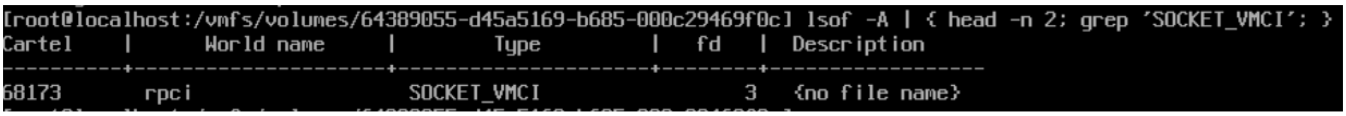


Figure 8: lsof -A showing process listening on a VMCI port

While VMCI communication is normally invisible to common network sniffing tools without modification, public research exists which documents how to sniff this traffic. Andra Paraschiv and Stefano Garzarella covered this and much more in their talk "Leveraging virtio-vsock in the cloud and containers" in FOSDEM 2021 (recording | slide deck, slide 23).

In addition to active VMCI backdoors, other historical artifacts can be found on ESXi hosts and guest machines that can be indicative of past threat actor activity. The following section describes some of the artifacts that arise from both successful compromise of an ESXi host using the vpxuser, and failed attempts to use Guest Operations on guest VMs without the exploit.

Additional Artifacts

Expected vpxuser Behavior on ESXi Hosts

Since the credentials for the `vpxuser` can be harvested on one (1) vCenter and result in the compromise of multiple connected ESXi hosts, it is important to understand the expected behavior of this user so any anomalous activity can be alerted on and investigated further.

Since the `vpxuser` account is a service account designed to administer ESXi hosts from vCenter servers, vCenter servers are the only place that this account should be expected to login from. Additionally, due to being a service account with automatic password rotation and with no expected user interaction, failed logons from the `vpxuser` should be considered abnormal and investigated further. An exception to this could potentially occur when an authentication check fails as the password is being rotated, however this should not occur often enough to cause a large amount of false positive alerts.

In addition to expected logon locations, there is only one user agent that the `vpxuser` should have when it logs onto an ESXi host as seen in the `hostd.log` examples in Figure 9. Normally, the expected user agent would be `VMware-client/X.X.X`. While investigating UNC3886 activity, Mandiant identified the threat actor connecting to ESXi hosts utilizing the following user agents:

- pyvmomi Python/3.5.5 (Linux; 4.4.157-1.ph1; x86_64)
- PowerCLI/12.5.0.19195797

VMware confirmed that the aforementioned user agents are not expected from the `vpxuser` when authenticating to an ESXi host and would be considered anomalous, warranting further investigation.

Figure 9: hostd.log entries showing abnormal vpxuser User Agents

```
info hostd[B981B70] [Originator@6876 sub=Vimsvc.ha-eventmgr opID=2dee888f] Event 488 : User
vpxuser@<vCenter IP> logged in as pyvmomi Python/3.5.5 (Linux; 4.4.157-1.ph1; x86_64)

info hostd[C481B70] [Originator@6876 sub=Vimsvc.ha-eventmgr opID=2dee88ad user=vpxuser] Event 491 : User
vpxuser@<vCenter IP> logged out (login time: <login time>, number of API invocations: 0, user agent:
pyvmomi Python/3.5.5 (Linux; 4.4.157-1.ph1; x86_64))

info hostd[11EC2B70] [Originator@6876 sub=Vimsvc.ha-eventmgr opID=bb47b2c8] Event 1461 : User
vpxuser@<vCenter IP> logged in as PowerCLI/12.5.0.19195797

info hostd[11E81B70] [Originator@6876 sub=Vimsvc.ha-eventmgr opID=bb47b37b user=vpxuser] Event 1462 : User
vpxuser@<vCenter IP> logged out (login time: <login time>, number of API invocations: 0, user agent:
PowerCLI/12.5.0.19195797)
```

In the situation where a `.vib` file is recovered, it is important to understand that `.vib` files are actually `AR` archives to be able to properly inspect them. To extract the contents of the VIB properly, the command `ar -xv <.vib File>` can be used.

In addition to abnormal vpxuser interactions, abnormal remote VIB installations can also shed light on past attacker activity, even if the VIBs have since been cleaned from the systems. The following section details additional log entries found in the `esxupdate.log` which can show where VIBs were remotely installed from.

Installation of VIBs Logs with download from vCenter

Within the `esxupdate.log`, a specific log entry will log the installation of a new VIB along with the `viburl` if it is installed remotely. Per Figure 10, the log entry provides the URL the VIB was downloaded from, the options used to install the VIB, and the destination that the VIB was downloaded to on the ESXi host.

Figure 10: VIBURL esxupdate.log entry

```
esxupdate: 376185: root: INFO: Options = {'nomaintmode': False, 'oktoremove': False, 'force': False,
'nosigcheck': True, 'noliveinstall': False, 'updateonly': False, 'viburl': , 'pending': None, 'profile':
None, 'proxy': None, 'nameid': None, 'depot': None, 'downgrade': None, 'dryrun': False, 'level': None}

esxupdate: 376185: downloader: DEBUG: Downloading http://<vCenter IP>:8080/ata-pata-pdc20279.vib to
/tmp/vibdownload/VMW_bootbank_ata-pata-pdc20279_1.0-3vmw.670.0.0.8169922.vib...
```

Looking for abnormal installation options such as `force/nosigcheck`, anomalous sources and VIB names, and suspicious ports used for installs can act as additional detection mechanisms.

Thus far, the blog post has covered logging for successful Guest Operations and abnormal vpxuser activity on ESXi hosts. The following section details how failed Guest Operations are logged on both ESXi hosts and guest VMs if the exploit is not properly executed and authentication challenges are made.

Failed Guest Operation events on ESXi Host

When Guest Operations fail, they are logged on guest machines in a consistent manner on the operating system, but failed Guest Operations can result in multiple types of error messages on the ESXi host that should be searched for within the `hostd.log`.

If the exploit is executed successfully, but the guest machine does not have vmtools installed, the hostd.log will log the error “vim.fault.GuestOperationsUnavailable” per Figure 11.

Figure 11: ESXi hostd.log vim.fault.GuestOperationsUnavailable

```
/var/log/hostd.log-2023-03-20T18:19:46.994Z info hostd[265567] [Originator@6876
sub=AdapterServer opID=216dd829 user=root] AdapterServer caught exception;
<<520eeb98-8189-0488-f079-b1b09e197338, <TCP '127.0.0.1 : 8307'>, <TCP '127.0.0.1 : 48405'>>,
ha-guest-operations-process-manager, vim.vm.guest.ProcessManager.startProgram>,
N3Vim5Fault26GuestOperationsUnavailable9ExceptionE(Fault cause:
vim.fault.GuestOperationsUnavailable

/var/log/hostd.log---> )

/var/log/hostd.log--->
[context]zKq7AVICAgAAAKEvNgEPaG9zdGQAACJDF2xpYnZtYWNvcmUuc28AATp37WxpYnZpbS10eXBlcy5zbwCBF8EHA
YGBYQgBgRYdBAECyO/HaG9zdGQAAkuvcAJmsHCBcAAVAQKKklIArMctADQDLgDiED8DO30AbGlicHRocmVhZC5zby4wA
ARt0Q5saWJjLnNvLjYA[/context]

/var/log/hostd.log-2023-03-20T18:19:46.999Z info hostd[265567] [Originator@6876 sub=Solo.Vmomi
opID=216dd829 user=root] Activation finished; <<520eeb98-8189-0488-f079-b1b09e197338, <TCP '127.0.0.1 :
8307'>, <TCP '127.0.0.1 : 48405'>>, ha-guest-operations-process-manager,
vim.vm.guest.ProcessManager.startProgram>

/var/log/hostd.log-2023-03-20T18:19:46.999Z verbose hostd[265567] [Originator@6876 sub=Solo.Vmomi
opID=216dd829 user=root] Arg vm:

/var/log/hostd.log---> 'vim.VirtualMachine:3'

/var/log/hostd.log-2023-03-20T18:19:46.999Z verbose hostd[265567] [Originator@6876 sub=Solo.Vmomi
opID=216dd829 user=root] Arg auth:

/var/log/hostd.log---> (vim.vm.guest.NamePasswordAuthentication) {

/var/log/hostd.log--->    interactiveSession = false,

/var/log/hostd.log:-->    username = "vinny",

/var/log/hostd.log--->    password = (not shown)

/var/log/hostd.log---> }

/var/log/hostd.log-2023-03-20T18:19:46.999Z verbose hostd[265567] [Originator@6876 sub=Solo.Vmomi
opID=216dd829 user=root] Arg spec:

/var/log/hostd.log---> (vim.vm.guest.ProcessManager.ProgramSpec) {

/var/log/hostd.log--->    programPath = "C:\Windows\system32\cmd.exe",

/var/log/hostd.log--->    arguments = "/c dir /od /s /a c:\ > C:\Windows\Temp\hi.txt",

/var/log/hostd.log---> }

/var/log/hostd.log-2023-03-20T18:19:46.999Z info hostd[265567] [Originator@6876 sub=Solo.Vmomi
opID=216dd829 user=root] Throw vim.fault.GuestOperationsUnavailable

/var/log/hostd.log-2023-03-20T18:19:46.999Z info hostd[265567] [Originator@6876 sub=Solo.Vmomi
opID=216dd829 user=root] Result:

Install vmtoolsd.exe on Windows Guest Machine
```

If vmtools are installed on the target guest machine but the exploit was not executed successfully, the host will validate whether the passed credentials are valid. When invalid credentials are used to attempt a Guest Operation, the error “vim.fault.InvalidGuestLogin” will be written to the hostd.log as seen in Figure 12.

Figure 12: ESXi hostd.log vim.fault.InvalidGuestLogin

```
/var/log/hostd.log-2023-03-20T18:57:17.574Z info hostd[265189] [Originator@6876 sub=Default opID=71a2fa8c]
Accepted password for user root from 127.0.0.1

/var/log/hostd.log-2023-03-20T18:57:17.574Z warning hostd[265189] [Originator@6876 sub=Vimsvc
opID=71a2fa8c] Refresh function is not configured.User data can't be added to scheduler.User name: root

/var/log/hostd.log-2023-03-20T18:57:17.574Z info hostd[265189] [Originator@6876 sub=Vimsvc.ha-eventmgr
opID=71a2fa8c] Event 110 : User root@127.0.0.1 logged in as pyvmomi Python/3.8.13 (VMkernel; 7.0.3; x86_64)

/var/log/hostd.log-2023-03-20T18:57:17.630Z info hostd[265191] [Originator@6876
sub=Vmsvc.vm:/vmfs/volumes/63e135fc-5c857e1b-5a1a-000c297c1baf/Victim-Win/Victim-Win.vmx opID=71a2fa94
user=root] State Transition (VM_STATE_ON -> VM_STATE_GUEST_OPERATION)
```

```
/var/log/hostd.log-2023-03-20T18:57:17.678Z verbose hostd[265191] [Originator@6876
sub=Vigor.Vmsvc.vm:/vmfs/volumes/63e135fc-5c857e1b-5a1a-000c297c1baf/Victim-Win/Victim-Win.vmx opID=esxui-
e1ff-fa23] Start a program translated error to vim.fault.InvalidGuestLogin

/var/log/hostd.log-2023-03-20T18:57:17.678Z verbose hostd[265191] [Originator@6876
sub=Vigor.Vmsvc.vm:/vmfs/volumes/63e135fc-5c857e1b-5a1a-000c297c1baf/Victim-Win/Victim-Win.vmx opID=esxui-
e1ff-fa23] Start a program message:

/var/log/hostd.log-2023-03-20T18:57:17.679Z info hostd[264228] [Originator@6876 sub=Vimsvc.ha-eventmgr
opID=esxui-e1ff-fa23] Event 111 : Guest operation authentication failed for operation Start Program on
Virtual machine Victim-Win.

/var/log/hostd.log-2023-03-20T18:57:17.681Z info hostd[264228] [Originator@6876
sub=Vmsvc.vm:/vmfs/volumes/63e135fc-5c857e1b-5a1a-000c297c1baf/Victim-Win/Victim-Win.vmx opID=esxui-e1ff-
fa23] State Transition (VM_STATE_GUEST_OPERATION -> VM_STATE_ON)

/var/log/hostd.log-2023-03-20T18:57:17.685Z info hostd[264228] [Originator@6876
sub=Vmsvc.vm:/vmfs/volumes/63e135fc-5c857e1b-5a1a-000c297c1baf/Victim-Win/Victim-Win.vmx opID=esxui-e1ff-
fa23] opCode=4 auth=<hidden> programPath=C:\Windows\system32\cmd.exe arguments=/c dir /od /s /a c:\ >
C:\Windows\Temp\hi.txt failed

/var/log/hostd.log-2023-03-20T18:57:17.685Z info hostd[264228] [Originator@6876 sub=Solo.Vmomi opID=esxui-
e1ff-fa23] Activation finished; <<5275a378-8cf9-dcd5-1916-c4c8387c1a02, <TCP '127.0.0.1 : 8307'>, <TCP
'127.0.0.1 : 30841'>>, ha-guest-operations-process-manager, vim.vm.guest.ProcessManager.startProgram>

/var/log/hostd.log-2023-03-20T18:57:17.685Z verbose hostd[264228] [Originator@6876 sub=Solo.Vmomi
opID=esxui-e1ff-fa23] Arg vm:

/var/log/hostd.log---> 'vim.VirtualMachine:3'

/var/log/hostd.log-2023-03-20T18:57:17.685Z verbose hostd[264228] [Originator@6876 sub=Solo.Vmomi
opID=esxui-e1ff-fa23] Arg auth:

/var/log/hostd.log---> (vim.vm.guest.NamePasswordAuthentication) {

/var/log/hostd.log--->    interactiveSession = false,

/var/log/hostd.log:-->    username = "vinny",

/var/log/hostd.log--->    password = (not shown)

/var/log/hostd.log---> }

/var/log/hostd.log-2023-03-20T18:57:17.685Z verbose hostd[264228] [Originator@6876 sub=Solo.Vmomi
opID=esxui-e1ff-fa23] Arg spec:

/var/log/hostd.log---> (vim.vm.guest.ProcessManager.ProgramSpec) {

/var/log/hostd.log--->    programPath = "C:\Windows\system32\cmd.exe",

/var/log/hostd.log--->    arguments = "/c dir /od /s /a c:\ > C:\Windows\Temp\hi.txt",

/var/log/hostd.log---> }

/var/log/hostd.log-2023-03-20T18:57:17.685Z info hostd[264228] [Originator@6876 sub=Solo.Vmomi opID=esxui-
e1ff-fa23] Throw vim.fault.InvalidGuestLogin

/var/log/hostd.log-2023-03-20T18:57:17.685Z info hostd[264228] [Originator@6876 sub=Solo.Vmomi opID=esxui-
e1ff-fa23] Result:

/var/log/hostd.log---> (vim.fault.InvalidGuestLogin) {

/var/log/hostd.log--->    msg = "",

/var/log/hostd.log---> }
```

When Guest Operations fail with vmtools installed on the target guest machine and due to invalid credentials, the guest machines will also log the failed checks. The following sections describe how both Windows and Linux machines log these failed validation attempts.

Failed Guest Operations on Linux Guests

If vmtoolsd is enabled on a Linux guest but the exploit was not executed correctly, an authentication check is made against a Linux guest machine. If this check fails, `/var/log/secure` will generate the event “Failed pam_unix(vmtoolsd:auth)” as seen in Figure 13 with the name of the user which was used to authenticate.

```
./secure-Apr 17 13:42:27 localhost polkitd[736]: Acquired the name org.freedesktop.PolicyKit1 on the
 system bus
./secure-Apr 17 13:42:33 localhost sshd[1038]: Server listening on 0.0.0.0 port 22.
./secure-Apr 17 13:42:33 localhost sshd[1038]: Server listening on :: port 22.
./secure-Apr 17 13:42:55 localhost login: pam_unix(login:session): session opened for user root by L
OGIN(uid=0)
./secure-Apr 17 13:42:55 localhost login: ROOT LOGIN ON tty1
./secure:Apr 17 13:44:47 localhost unix_chkpwd[1354]: password check failed for user (root)
./secure-Apr 17 13:44:47 localhost VGAuth[744]: pam_unix(vmtoolsd:auth): authentication failure; log
name= uid=0 euid=0 tty= ruser= rhost=  user=root
./secure:Apr 17 13:44:47 localhost unix_chkpwd[1355]: password check failed for user (root)
./secure-Apr 17 13:44:50 localhost VGAuth[744]: PAM 1 more authentication failure; logname= uid=0 eu
id=0 tty= ruser= rhost=  user=root
./secure-Apr 17 13:44:50 localhost VGAuth[744]: vmtoolsd: Username and password mismatch for 'root'
```

Figure 13: Linux vmtoolsd failed authentication

If the debugging logging level for vmsvc logging is enabled on the Linux machine, the failed impersonation can be observed along with the guest operation type that failed to run as seen in Figure 14.

```
./vmsvc.log-[Apr 17 13:44:47.501] [   debug] [VCGA] Unable to open './messages/en/VGAuthLib.vmsg': N
o such file or directory
./vmsvc.log-[Apr 17 13:44:47.501] [ message] [VCGA] VGAuth 'build-5055683' initialized for applicati
on 'vmtoolsd'.  Context created at 0x5564f9f06cf0
./vmsvc.log-[Apr 17 13:44:47.507] [ message] [VCGA] PAM up and running.
./vmsvc.log-[Apr 17 13:44:50.108] [ warning] [VCGA] PAM error: Authentication failure (7), mapped to
 VGAuth error 12
./vmsvc.log-[Apr 17 13:44:50.108] [   debug] [vix] VixToolsTranslateVGAuthError: translated VGAuth e
rr 0xc to Vix err 3050
./vmsvc.log:[Apr 17 13:44:50.108] [ warning] [vix] VixToolsImpersonateUser: impersonation failed (30
50)
./vmsvc.log-[Apr 17 13:44:50.108] [ message] [vix] VixToolsInitiateFileTransferToGuest: opcode 189 r
eturning 3050
./vmsvc.log-[Apr 17 13:44:50.108] [ message] [vix] ToolsDaemonTcloReceiveVixCommand: command 189, ad
ditionalError = 4294967295
./vmsvc.log-[Apr 17 13:44:50.108] [   debug] [vmsvc] RpcIn: sending 12 bytes
./vmsvc.log-[Apr 17 13:44:50.110] [   debug] [vmsvc] RpcIn: received 5 bytes, content:"ping\00"
./vmsvc.log-[Apr 17 13:44:50.110] [   debug] [vmsvc] RpcIn: sending 3 bytes
[root@localhost log]# _
```

Figure 14: Linux failed authentication vmsvc.log

Failed Guest Operations on Windows Guests

Similar to Linux guest machines, when vmtoolsd is enabled but the exploit is not executed, an authentication check is made against a Windows guest machine resulting in three 4625 events in the Windows Security Event Logs. These three events occur in quick succession with one Type 2, one Type 4, and one Type 5 logon attempt (Windows Logon Types). Additionally, the following attributes are consistent amongst the 4625 events:

- The account name is the username passed from the host that failed the check
- The Caller Process Name is `C:\Program Files\VMware\VMware Tools\vmtoolsd.exe`
- The `Network Information: Workstation Name` will be the Windows Guest Hostname.

Alerting and detections are useful when reacting to a threat actor already in the environment, but to be proactive in protecting against potential incidents before they occur, proper hardening and configuration of virtualization technologies is required. The rest of this blog post will describe the many layers of protections that can be put in place at the vCenter, ESXi host, and guest machine level to help protect against unauthorized access to these technologies.

Containment and Hardening Recommendations

Network Segmentation of Administrative Interfaces for ESXi Hosts and vCenter Servers

To gain access to ESXi hosts and vCenter Servers, attackers need to be able to reach the administrative interfaces of these systems. Network segmentation provides a significant opportunity for reducing the overall attack surface and presents an attacker with roadblocks throughout each step of the attack lifecycle. Isolating the administrative interface to a separate network segment

from the main production network has the added benefit of helping to contain compromises of other systems and infrastructure from spreading to the VMware hypervisor infrastructure.

Ensure that any VMKernel interfaces that are configured for ESXi Management, vMotion, and vSAN are deployed only in a restricted network segment, and that any supporting systems that are needed for those functions are also deployed in the restricted network segment. In addition to isolating the ESXi and vCenter Server administrative interfaces to a separate VLAN, restrict access to only specifically allowed, privileged management workstations. Ensure that these management workstations are implementing best practices, such as multi-factor authentication, Windows Defender Credential Guard, Remote Credential Guard, and Application Control.

Firewall Restrictions for Administrative Access

Configure the ESXi hosts and vCenter Servers to only allow connections on their administrative interfaces from the IP addresses of the privileged management workstations mentioned in the previous section. Ensure that these firewall configurations apply to each service utilized by the server, such as SSH, vSphere Web Access, etc. Alternatively, utilize a network firewall appliance in front of the ESXi and vCenter Server administrative interfaces and implement the IP address restrictions there.

Figure 15 shows the PowerCLI command that can be used to list all services on ESXi host(s).

Figure 15: PowerCLI command to list all services

```
PS C:\> Get-VMHostService -VMHost 192.168.1.10

Key                  Label                          Policy     Running  Required
---                  -----                          ------     -------  --------
DCUI                 Direct Console UI              on         True     False
TSM                  ESXi Shell                     on         True     False
TSM-SSH              SSH                            on         True     False
attestd              attestd                        off        False    False
dpd                  dpd                            off        False    False
kmxd                 kmxd                           off        False    False
lbtd                 Load-Based Teaming Daemon      on         True     False
lwsmd                Active Directory Service       off        False    False
ntpd                 NTP Daemon                     on         True     False
pcscd                PC/SC Smart Card Daemon        off        False    False
ptpd                 PTP Daemon                     off        False    False
sfcbd-watchdog       CIM Server                     on         False    False
slpd                 slpd                           off        False    False
snmpd                SNMP Server                    on         False    False
vltd                 vltd                           off        False    False
vmsyslogd            Syslog Server                  on         True     True
vpxa                 Vmware vCenter Agent           on         False    False
xorg                 X.Org Server                   on         False    False
```

The firewall on ESXi hosts is enabled by default and can be managed from the vSphere client web UI, the ESXCLI command shell, and PowerCLI. By default, the firewall is configured to block both incoming and outgoing network traffic. Services that are enabled in the host's security profile are allowed through the host firewall and are configured to allow access for all IP addresses. To further harden access to services, configure the ESXi firewall to allow access only from the IP addresses of privileged management workstations.

Figure 16 shows the PowerCLI command that retrieves and lists the ESXi host firewall exceptions.

Figure 16: PowerCLI command to list all firewall exceptions

```
PS C:\> Get-VMHostFirewallException -VMHost 192.168.1.10
Name                 Enabled IncomingPorts  OutgoingPorts  Protocols  ServiceRunning
----                 ------- -------------  -------------  ---------  --------------
CIM Server           True    5988                          TCP        False
CIM Secure Server    True    5989                          TCP        False
CIM SLP              True    427            427            UDP, TCP   False
DHCPv6               True    546            547            TCP, UDP
DVFilter             False   2222                          TCP
DVSSync              True    8301, 8302     8302, 8301     UDP
HBR                  True                   31031, 44046   TCP
NFC                  True    902            902            TCP
WOL                  True                   9              UDP
Active Directory All False   2020           88, 123, 13... UDP, TCP
vSAN Clustering S... False   12345, 2345... 12345, 2345... UDP
DHCP Client          True    68             68             UDP
DNS Client           True                   53             UDP, TCP
esxio-orchestrator   False   8084                          TCP
esxupdate            False                  443            TCP
etcdClientComm       False   2379           2379           TCP
etcdPeerComm         False   2380           2380           TCP
Fault Tolerance      True    8300           80, 8300       TCP
FTP Client           False   20             21             TCP
gdbserver            False   1000-9999, ...                TCP
gstored              False                  443            TCP
httpClient           False                  80, 443        TCP
Software iSCSI Cl... False                  3260           TCP
iofiltervp           True    9080                          TCP
NSX Distributed L... False   6999           6999           UDP
iwarp-pm             False   3935           3935           UDP
nfs41Client          False                  0-65535        TCP
NFS Client           False                  0-65535        TCP
NTP Client           True                   123            UDP        True
nvmetcp              False                  8009, 4420     TCP
PTP Client           False   319-320        319-320        UDP        False
pvrdma               False   28250-28761    28250-28761    TCP
vSAN Transport       False   2233, 12443    2233, 12443    TCP
VM serial port co... False   23, 1024-65535 0-65535        TCP
settingsd            False   8083           8083           TCP
SNMP Server          True    161                           UDP        False
```

```
SSH Client           False                 22             TCP
SSH Server           True    22                           TCP
syslog               False                 514, 1514      UDP, TCP
trusted-infrastru... True                  0-65535        TCP
trusted-infrastru... False                 0-65535        TCP
vCenter Update Ma... True                  80, 9000-9100  TCP
vMotion              True    8000          8000           TCP
VM serial port co... False                 0-65535        TCP
vSphereCCP           False   81, 444, 20... 81, 444, 20... TCP
vSphere Web Client   True    902, 443                     TCP
vdfs                 False   1564          1564           TCP
vic-engine           False                 2377           TCP
vit                  False   3260                         TCP
vltd                 False                 1492           TCP        False
VMware vCenter Agent True                  902            UDP        False
vsanEncryption       False                 0-65535        TCP
vsanhealth-unicas... False   5201          5201           UDP, TCP
vvold                False                 0-65535        TCP
vSphere Web Access   True    80                           TCP
```

For additional information on ESXi Firewall Configuration, reference VMware's ESXi page.

Sign On Identity Sources

At the time of this blog post, it is unknown how the attackers initially gained privileged access to the vCenter Server. Mandiant recommends reviewing current identity source configurations to enforce the mindset of least privilege for access to ESXi hosts and vCenter Servers.

Identity sources on vCenter Servers dictate which external services (if any) vCenter will utilize to authorize administrative users to the vCenter Server. Possible identity sources include native Microsoft Active Directory Domains and OpenLDAP directory services. Microsoft Active Directory can be configured to utilize Integrated Windows Authentication or LDAP.

Mandiant has observed threat actors discovering the presence of ESXi hosts and vCenter Servers that are integrated with Microsoft Active Directory and actively targeting the administrative accounts to infiltrate the VMware hypervisor infrastructure. Consider decoupling ESXi and vCenter Servers from Active Directory and use vCenter Single Sign-On. Removing ESXi and vCenter from Active Directory will prevent any compromised Active Directory accounts from being able to be used to authenticate directly to the virtualization infrastructure.

If Active Directory is utilized as an identity source, Mandiant recommends actively monitoring the groups and/or users that are allowed to perform ESXi and vCenter administrative capabilities for changes/modifications/additions.

For additional information on configuring vSphere authentication please see VMware's documentation regarding vCenter single sign-on.

Multi-Factor Authentication (MFA) for Admins

Attackers will commonly target the vCenter Server Appliance and leverage single-factor authentication for accessing applications with legitimate or stolen credentials. Using MFA reduces the risk of an attacker gaining access to highly privileged systems and applications.

Mandiant recommends enforcing MFA to access all vCenter Server instances. Some supported methods of utilizing multifactor authentication with vCenter Server include smart card authentication, RSA SecureID Authentication, and Duo Security. For additional information related to MFA integration, please reference this VMware article.

Disable Remote SSH

Attackers have often targeted and utilized SSH access on ESXi and vCenter hosts to gain access and propagate malware throughout the environment. Eliminating the exposure of open ports reduces the ability for an attacker to leverage this method of access. Some management solutions require the use of SSH to accomplish their automated tasks. If one of these solutions is being utilized, SSH access can be secured by using an allow list to limit appropriate source IP addresses that are used by the management solution.

Figure 17 shows the PowerCLI command that can be used to list the status of SSH on ESXi host(s).

Figure 17: PowerCLI command to list status of SSH service

```
PS C:\> Get-VMHostService -VMHost 192.168.1.10 | Where-Object {$_.Key -match “SSH”}

Key                  Label                          Policy     Running  Required

---                  -----                          ------     -------  --------

TSM-SSH              SSH                            on         True     False
```

Enable Lockdown Mode

Requiring all access to occur through the vCenter Server can reduce the risk of an attacker bypassing access controls and accessing the ESXi host directly, then leveraging that access to elevate privileges or perform other malicious activity.

Enabling Lockdown Mode disables direct access to an ESXi host and requires that the host be managed remotely using vCenter. This is done to ensure the roles and access controls implemented in vCenter are always enforced and users cannot bypass them by logging into a host directly. By forcing all interaction to occur through vCenter, the risk of someone inadvertently attaining elevated privileges or performing tasks that are not properly audited is greatly reduced.

To reduce the risk and exposure of credentials correlating to accounts with access to vCenter, it is recommended that unique and dedicated accounts be used for this access. If vCenter must be joined to an on-premises Active Directory (AD) domain and Integrated Windows Authentication is configured, the scope of accounts provided access to vCenter should be restricted to a small subset, where the account(s) are not co-utilized for performing interactive or remote logons for endpoints.

Figure 18 shows the PowerCLI command that can be used to determine the Lockdown status of an ESXi host.

Figure 18: PowerCLI command to show LockdownMode status

```
PS C:\> (Get-VMHost 192.168.1.10).ExtensionData.Config.LockdownMode

lockdownDisabled
```

Enforcing VIB Acceptance Levels & Conducting VIB Verification

Configuring a more secure acceptance level will restrict the installation of unsigned VIBs to protect the security and integrity of an ESXi host. Determine the appropriate risk acceptance level for vSphere Installable Bundles (VIBs) and enforce acceptance levels in the Security Profiles for ESXi hosts. This protects the integrity of the hosts and ensures unsigned VIBs cannot be installed.

VMware and Mandiant recommend not allowing users to install unsigned (community-supported) VIBs. Concurrently with Mandiant’s disclosure of threat actors utilizing unsigned VIBs to install backdoors on compromised ESXi hosts via the Bad VIB(E) Part One and Part Two blog posts in September 2022, VMware published the Mitigation and Threat Hunting Guidance for Unsigned vSphere Installation Bundles (VIBs) in ESXi article. This VMware article includes a PowerCLI script that can be utilized to audit ESXi hosts for unsigned VIBs.

Finding unsigned VIBs on ESXi hosts is not definitive proof of a compromise. Mandiant and VMware recommend organizations investigate the origin of any installed unsigned VIBs and if a compromise is suspected, follow their established incident response procedures.

Prevent Execution of Unsigned Binaries and Enable Secure Boot

Enabling the `execInstalledOnly` feature in ESXi will restrict unsigned binaries from being executed in the ESXi host and can ensure that only signed binaries are allowed. This can significantly reduce the risk of unknown executable files being executed on ESXi hosts. This article by VMware provides additional information to enable the `execInstalledOnly` feature.

Mandiant recommends enabling `execInstalledOnly` enforcement using ESXCLI to change the Trusted Platform Module (TPM) configuration settings in ESXi hosts. This will enforce integrity checks of vSphere Installable Bundles (VIBs), governed by the configured acceptance level. Instructing ESXi to only execute binaries that originated from a valid VIB installed on the host makes it harder for threat actors to use prebuilt binaries during a compromise. UEFI secure boot needs to be enabled before enforcing `execInstalledOnly` settings. For additional information for enabling UEFI Secure boot, please reference the following VMware article.

Centralize VMware Logging to SIEM Solution

Insufficient logging can hinder a successful response to an incident, as there may be limited evidence as to actions taken that led to a compromise. Without a unified approach to centralized logging, organizations will encounter difficulties detecting and responding to threats within the environment.

ESXi and vCenter Server record host activity in log files using a syslog facility and can be configured to store log files on an in-memory file system (which means logs will not persist reboots) or to a permanent datastore. Mandiant recommends that all ESXi and vCenter Server host logging should always be configured to a persistent datastore. The following VMware articles cover the locations of log locations on both vCenter and ESXi.

Remote logging to a central log collector or SIEM provides a secure, centralized store for vSphere logs. Having a centralized location for logs enables detection and threat hunting at scale. Mandiant recommends that all logs for critical assets be sent to a centralized system, and that high-fidelity alerts and alert logic be configured to alert on anomalous activity. Logging to a secure, centralized logging system also helps prevent log tampering and is also a long-term audit record. For information on Configuring Syslog on ESXi Hosts and ESXi Syslog Options, please reference the linked VMware articles.

Utilize VMware's vSphere Security Configuration Guide

VMware has created Security Configuration Guides for specific versions of VMware products, including vSphere. These guides provide VMware's recommended security baselines in an easy-to-read spreadsheet format with example PowerCLI commands to configure security features. VMware's Security Configuration and Hardening Guides are available now.

Conclusion

As attackers continue to find new techniques to evade detection and technologies to persist on, defenders must continue to evolve as well. To be able to identify when advanced threat actors are present in the environment, defenders need to not only focus on building out a workflow that alerts on known indicators, but also has a strong baseline understanding of expected activity to notify when abnormalities occur. This process is not an easy or quick one to implement, but the benefits achieved in establishing this system are twofold. In addition to greater visibility, knowing and documenting your own environment to this degree enables streamlined communication and quicker response times when attackers do make their way through whatever preventative measures are in place.

Acknowledgements

Special thanks to Brad Slaybaugh, Jeremy Koppen, DJ Palombo, Joshua Kim, Matthew Maczko, Maegan Palombo, Rufus Brown, and Charles Carmakal for their assistance with the investigation and technical review of this blog post. In addition, we would also like to thank VMware for their collaboration and assistance with this research.