使用PowerShell创建、强化和验证ONTAP网络存储

NetApp Solutions

NetApp
September 26, 2024

目录

使用PowerShell创建、强化和验证ONTAP网络存储

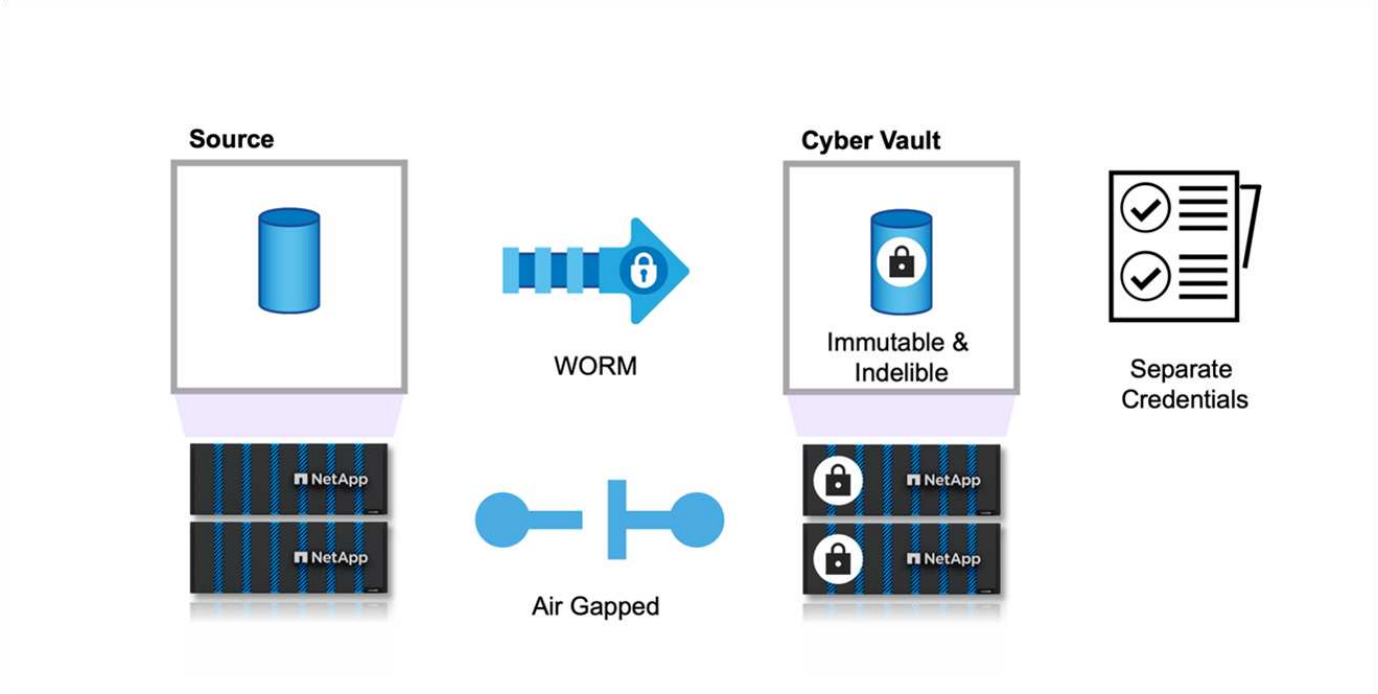
使用PowerShell的ONTAP网络存储概述

在当今的数字化环境中、保护企业的关键数据资产不仅是一项最佳实践、而且也是一项业务要务。网络威胁以前所未有的速度不断发展、传统数据保护措施已不再足以保证敏感信息的安全。这正是网络存储的出现之处。NetApp基于ONTAP的尖端解决方案将先进的空气封顶技术与强大的数据保护措施相结合、为抵御网络威胁创建了一个不可阻挡的屏障。通过使用安全强化技术隔离最有价值的数据、网络存储可最大限度地减少攻击面、从而使最关键的数据保持机密性、完整性、并在需要时随时可用。

网络存储是一种由多层保护(如防火墙、网络和存储)组成的安全存储设施。这些组件可保护关键业务运营所需的重要恢复数据。网络存储的组件会根据存储策略定期与基本生产数据同步、但在其他方面仍无法访问。这种隔离且互不关联的设置可确保在发生网络攻击损害生产环境时、可以轻松地从网络存储执行可靠的最终恢复。

NetApp通过配置网络、禁用LIFs、更新防火墙规则以及将系统与外部网络和Internet隔离、可以轻松地为网络存储创建空隙。这种强大的方法可以有效地将系统与外部网络和互联网断开连接、从而提供无与伦比的保护、防止远程网络攻击和未经授权的访问尝试、使系统免受基于网络的威胁和入侵的影响。

将此功能与SnapLock Compliance保护相结合、即使ONTAP管理员或NetApp支持人员也无法修改或删除数据。SnapLock会根据SEC和金融监管局的法规定期进行审核、以确保数据故障恢复能力满足银行业这些严格的WORM和数据保留法规。NetApp是唯一经过NSA CSFC验证可存储顶级机密数据的企业级存储。

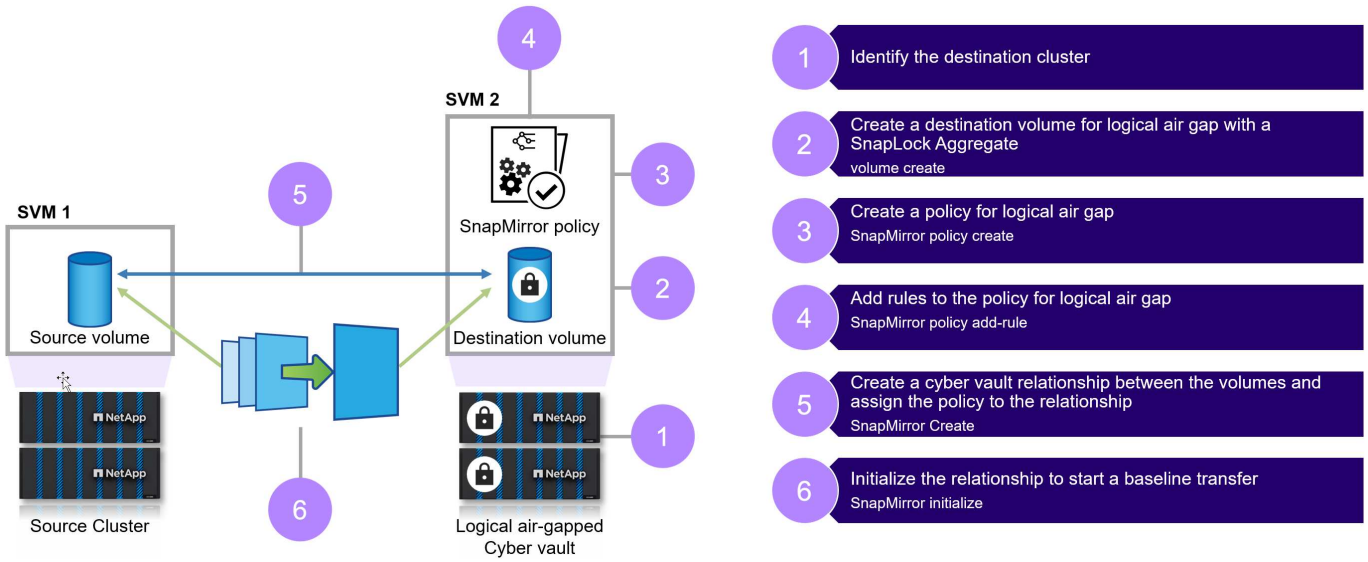


本文档介绍如何将用于内部ONTAP存储的NetApp网络存储自动配置到另一个指定的ONTAP存储、并使用不可修改的快照增加一层保护、以防止网络攻击增加、从而实现快速恢复。在此架构中、将根据ONTAP最佳实践应用整个配置。最后一节介绍了在发生攻击时执行恢复的说明。

使用FSx for ONTAP在AWS中创建指定的网络存储也适用相同的解决方案。

创建ONTAP网络存储的高级步骤

- 创建对等关系
  - 使用ONTAP存储的生产站点与指定的网络存储ONTAP存储建立对等关系
- 创建SnapLock Compliance卷
- 设置SnapMirror关系和规则以设置标签
  - 已配置SnapMirror关系和相应的计划
- 在启动SnapMirror (存储)传输之前设置保留
  - 对复制的数据应用保留锁定、从而进一步防止数据发生任何内部或数据故障。使用此选项时、无法在保留期限到期之前删除数据
  - 组织可以根据需要将这些数据保留几周/几个月
- 根据标签初始化SnapMirror关系
  - 初始传播和增量永久传输根据SnapMirror计划进行
  - 数据通过SnapLock Compliance进行保护(不可变、不可变)、并且数据可供恢复
- 实施严格的数据传输控制
  - 网络存储在有限的时间内与生产站点中的数据解除锁定、并与存储中的数据同步。传输完成后、连接将断开、关闭并再次锁定
- 快速恢复
  - 如果主站点在生产站点中受到影响、则网络存储中的数据将安全地恢复到原始生产环境或其他选定环境



解决方案组件

源集群和目标集群上的NetApp ONTAP 9.15.1。

ONTAP One：NetApp ONTAP的一体化许可证。

ONTAP One许可证使用的功能：

- SnapLock Compliance
- SnapMirror
- 多管理员验证
- ONTAP提供的所有强化功能
- 网络存储的单独RBAC凭据

所有ONTAP统一物理阵列均可用于网络存储、但基于AFF C系列容量的闪存系统和FAS混合闪存系统是实现此目的最经济高效的理想平台。"ONTAP网络存储规模估算"有关规模估算指导、请参见。

使用PowerShell创建ONTAP网络存储

使用传统方法的空载备份涉及到创建空间以及物理分离主介质和二级介质。通过将介质移至异地和/或断开连接、不良行为者将无法访问数据。这样可以保护数据、但可能会导致恢复时间变慢。使用SnapLock Compliance时、不需要进行物理隔离。SnapLock Compliance可保护存储的快照时间点只读副本、从而使数据能够快速访问、安全地不会被删除或不可删除、也不会被修改或不可变。

前提条件

开始执行本文档下一节中的步骤之前、请确保满足以下前提条件：

- 源集群必须运行ONTAP 9或更高版本。
- 源聚合和目标聚合必须为 64 位。
- 源集群和目标集群必须建立对等关系。
- 源和目标SVM必须建立对等关系。
- 确保已启用集群对等加密。

设置到ONTAP网络存储的数据传输需要几个步骤。在主卷上、使用适当的计划配置一个快照策略来指定要创建的副本以及创建时间、并分配标签以指定应由SnapVault传输的副本。在二级系统上、必须创建SnapMirror策略、指定要传输的Snapshot副本的标签以及应在网络存储上保留这些副本的数量。配置这些策略后、创建SnapVault关系并建立传输计划。

本文档假设已设置和配置主存储和指定的ONTAP网络存储。

网络存储集群可以与源数据位于同一数据中心、也可以位于不同数据中心。

创建ONTAP网络存储的步骤

1. 使用ONTAP命令行界面或系统管理器初始化Compliance时钟。
2. 创建启用了SnapLock Compliance的数据保护卷。
3. 使用SnapMirror create命令创建SnapVault数据保护关系。
4. 设置目标卷的默认SnapLock Compliance保留期限。

默认保留为"设置为最小值"。作为存储目标的 SnapLock 卷会为其分配默认保留期限。对于SnapLock Compliance卷、此时间段的值最初设置为最小0年、最大30年。首次提交每个 NetApp Snapshot 副本时都会使用此默认保留期限。保留期限可以稍后根据需要延长、但不能缩短。

上述步骤包括手动步骤。安全专家建议自动执行此过程、以避免因手动管理而产生较大的错误空间。以下代码段可完全自动执行SnapLock Compliance的前提条件和配置以及时钟初始化。

以下是用于初始化ONTAP Compliance时钟的PowerShell代码示例。

```
function initializeSnapLockComplianceClock {
    try {
        $nodes = Get-NcNode

        $isInitialized = $false
        logMessage -message "Cheking if snaplock compliance clock is
initialized"
        foreach($node in $nodes) {
            $check = Get-NcSnaplockComplianceClock -Node $node.Node
            if ($check.SnaplockComplianceClockSpecified -eq "True") {
                $isInitialized = $true
            }
        }

        if ($isInitialized) {
            logMessage -message "SnapLock Compliance clock already
initialized" -type "SUCCESS"
        } else {
            logMessage -message "Initializing SnapLock compliance clock"
            foreach($node in $nodes) {
                Set-NcSnaplockComplianceClock -Node $node.Node
            }
            logMessage -message "Successfully initialized SnapLock
Compliance clock" -type "SUCCESS"
        }
    } catch {
        handleError -errorMessage $_.Exception.Message
    }
}
```

以下是用于配置ONTAP网络存储的PowerShell代码示例。

```
function configureCyberVault {
    for($i = 0; $i -lt $DESTINATION_VOLUME_NAMES.Length; $i++) {
        try {
```

```
            # checking if the volume already exists and is of type
snaplock compliance
            logMessage -message "Checking if SnapLock Compliance volume
$($DESTINATION_VOLUME_NAMES[$i]) already exists in vServer
$DESTINATION_VSERVER"
            $volume = Get-NcVol -Vserver $DESTINATION_VSERVER -Volume
$DESTINATION_VOLUME_NAMES[$i] | Select-Object -Property Name, State,
TotalSize, Aggregate, Vserver, Snaplock | Where-Object { $_.Snaplock.Type
-eq "compliance" }
            if($volume) {
                $volume
                logMessage -message "SnapLock Compliance volume
$($DESTINATION_VOLUME_NAMES[$i]) already exists in vServer
$DESTINATION_VSERVER" -type "SUCCESS"
            } else {
                # Create SnapLock Compliance volume
                logMessage -message "Creating SnapLock Compliance volume:
$($DESTINATION_VOLUME_NAMES[$i])"
                New-NcVol -Name $DESTINATION_VOLUME_NAMES[$i] -Aggregate
$DESTINATION_AGGREGATE_NAMES[$i] -SnaplockType Compliance -Type DP -Size
$DESTINATION_VOLUME_SIZES[$i] -ErrorAction Stop | Select-Object -Property
Name, State, TotalSize, Aggregate, Vserver
                logMessage -message "Volume $($DESTINATION_VOLUME_NAMES[
$i]) created successfully" -type "SUCCESS"
            }

            # Set SnapLock volume attributes
            logMessage -message "Setting SnapLock volume attributes for
volume: $($DESTINATION_VOLUME_NAMES[$i])"
            Set-NcSnaplockVolAttr -Volume $DESTINATION_VOLUME_NAMES[$i]
-MinimumRetentionPeriod $SNAPLOCK_MIN_RETENTION -MaximumRetentionPeriod
$SNAPLOCK_MAX_RETENTION -ErrorAction Stop | Select-Object -Property Type,
MinimumRetentionPeriod, MaximumRetentionPeriod
            logMessage -message "SnapLock volume attributes set
successfully for volume: $($DESTINATION_VOLUME_NAMES[$i])" -type "SUCCESS"

            # checking snapmirror relationship
            logMessage -message "Checking if SnapMirror relationship
exists between source volume $($SOURCE_VOLUME_NAMES[$i]) and destination
SnapLock Compliance volume $($DESTINATION_VOLUME_NAMES[$i])"
            $snapmirror = Get-NcSnapmirror | Select-Object SourceCluster,
SourceLocation, DestinationCluster, DestinationLocation, Status,
MirrorState | Where-Object { $_.SourceCluster -eq
$SOURCE_ONTAP_CLUSTER_NAME -and $_.SourceLocation -eq "$($SOURCE_VSERVER)
:$($SOURCE_VOLUME_NAMES[$i])" -and $_.DestinationCluster -eq
$DESTINATION_ONTAP_CLUSTER_NAME -and $_.DestinationLocation -eq "
```

```
$($DESTINATION_VSERVER):$($DESTINATION_VOLUME_NAMES[$i])" -and ($_.Status
-eq "snapmirrored" -or $_.Status -eq "uninitialized") }
            if($snapmirror) {
                $snapmirror
                logMessage -message "SnapMirror relationship already
exists for volume: $($DESTINATION_VOLUME_NAMES[$i])" -type "SUCCESS"
            } else {
                # Create SnapMirror relationship
                logMessage -message "Creating SnapMirror relationship for
volume: $($DESTINATION_VOLUME_NAMES[$i])"
                New-NcSnapmirror -SourceCluster $SOURCE_ONTAP_CLUSTER_NAME
-SourceVserver $SOURCE_VSERVER -SourceVolume $SOURCE_VOLUME_NAMES[$i]
-DestinationCluster $DESTINATION_ONTAP_CLUSTER_NAME -DestinationVserver
$DESTINATION_VSERVER -DestinationVolume $DESTINATION_VOLUME_NAMES[$i]
-Policy $SNAPMIRROR_PROTECTION_POLICY -Schedule $SNAPMIRROR_SCHEDULE
-ErrorAction Stop | Select-Object -Property SourceCluster, SourceLocation,
DestinationCluster, DestinationLocation, Status, Policy, Schedule
                logMessage -message "SnapMirror relationship created
successfully for volume: $($DESTINATION_VOLUME_NAMES[$i])" -type "SUCCESS"
            }

        } catch {
            handleError -errorMessage $_.Exception.Message
        }
    }
}
```

1. 完成上述步骤后、使用SnapLock Compliance和SnapVault的气隙网络存储即已准备就绪。

在将快照数据传输到网络存储之前、必须初始化SnapVault关系。但是、在此之前、必须执行安全强化以保护存储。

使用PowerShell强化ONTAP网络存储

与传统解决方案相比、ONTAP网络存储可提供更强的抵御网络攻击的能力。在设计架构以增强安全性时、考虑减少攻击面积的措施至关重要。这可以通过各种方法来实现、例如实施强化密码策略、启用RBAC、锁定默认用户帐户、配置防火墙以及对存储系统进行任何更改时利用审批流。此外、限制特定IP地址的网络访问协议有助于限制潜在的漏洞。

ONTAP提供了一组控件、用于加强ONTAP存储。使用"ONTAP的指导和配置设置"帮助组织满足信息系统机密性、完整性和可用性方面的规定安全目标。

强化最佳实践

手动步骤

1. 创建具有预定义和自定义管理角色的指定用户。
2. 创建新的IP空间以隔离网络流量。
3. 创建驻留在新IP空间中的新SVM。
4. 确保正确配置防火墙路由策略、并根据需要定期审核和更新所有规则。

ONTAP命令行界面或通过自动化脚本

1. 通过多管理员验证(Multi-Admin Verification、MFA)保护管理
2. 为集群之间"传输中"的标准数据启用加密。
3. 使用强加密密码保护SSH并强制实施安全密码。
4. 启用全局FIPS。
5. 应禁用Telnet和远程Shell (RSH)。
6. 锁定默认管理员帐户。
7. 禁用数据BIFs并保护远程访问点的安全。
8. 禁用并删除未使用或无关的协议和服务。
9. 对网络流量进行加密。
10. 在设置超级用户和管理角色时、请使用最小特权原则。
11. 使用允许的IP选项限制HTTPS和SSH的特定IP地址。
12. 根据传输计划暂停和恢复复制。

要点1-4需要手动干预、例如指定一个隔离的网络、隔离IP空间等、并且需要事先执行。有关配置强化的详细信息，请参见"ONTAP安全强化指南"。其余部分可以轻松实现自动化、便于部署和监控。此协调方法的目标是提供一种机制来自动执行强化步骤、以使存储控制器适应未来需求。网络存储空隙开放的时间范围尽可能短。SnapVault利用增量永久技术、该技术只会将自上次更新以来的更改移至网络存储、从而最大程度地减少网络存储必须保持打开状态的时间。为了进一步优化工作流、网络存储的打开与复制计划相协调、以确保连接窗口最小。

以下是用于加密ONTAP控制器的PowerShell代码示例。

```
function removeSvmDataProtocols {
    try {

        # checking NFS service is disabled
        logMessage -message "Checking if NFS service is disabled on
vServer $DESTINATION_VSERVER"
        $nfsService = Get-NcNfsService
        if($nfsService) {
            # Remove NFS
            logMessage -message "Removing NFS protocol on vServer :
$DESTINATION_VSERVER"
            Remove-NcNfsService -VserverContext $DESTINATION_VSERVER
```

```
-Confirm:$false
            logMessage -message "NFS protocol removed on vServer :
$DESTINATION_VSERVER" -type "SUCCESS"
        } else {
            logMessage -message "NFS service is disabled on vServer
$DESTINATION_VSERVER" -type "SUCCESS"
        }

        # checking CIFS/SMB server is disabled
        logMessage -message "Checking if CIFS/SMB server is disabled on
vServer $DESTINATION_VSERVER"
        $cifsServer = Get-NcCifsServer
        if($cifsServer) {
            # Remove SMB/CIFS
            logMessage -message "Removing SMB/CIFS protocol on vServer :
$DESTINATION_VSERVER"
            $domainAdministratorUsername = Read-Host -Prompt "Enter Domain
administrator username"
            $domainAdministratorPassword = Read-Host -Prompt "Enter Domain
administrator password" -AsSecureString
            $plainPassword = [Runtime.InteropServices.Marshal
]::PtrToStringAuto([Runtime.InteropServices.Marshal]::SecureStringToBSTR($
domainAdministratorPassword))
            Remove-NcCifsServer -VserverContext $DESTINATION_VSERVER
-AdminUsername $domainAdministratorUsername -AdminPassword $plainPassword
-Confirm:$false -ErrorAction Stop
            logMessage -message "SMB/CIFS protocol removed on vServer :
$DESTINATION_VSERVER" -type "SUCCESS"
        } else {
            logMessage -message "CIFS/SMB server is disabled on vServer
$DESTINATION_VSERVER" -type "SUCCESS"
        }

        # checking iSCSI service is disabled
        logMessage -message "Checking if iSCSI service is disabled on
vServer $DESTINATION_VSERVER"
        $iscsiService = Get-NcIscsiService
        if($iscsiService) {
            # Remove iSCSI
            logMessage -message "Removing iSCSI protocol on vServer :
$DESTINATION_VSERVER"
            Remove-NcIscsiService -VserverContext $DESTINATION_VSERVER
-Confirm:$false
            logMessage -message "iSCSI protocol removed on vServer :
$DESTINATION_VSERVER" -type "SUCCESS"
        } else {
```

```
            logMessage -message "iSCSI service is disabled on vServer
$DESTINATION_VSERVER" -type "SUCCESS"
        }

        # checking FCP service is disabled
        logMessage -message "Checking if FCP service is disabled on
vServer $DESTINATION_VSERVER"
        $fcpService = Get-NcFcpService
        if($fcpService) {
            # Remove FCP
            logMessage -message "Removing FC protocol on vServer :
$DESTINATION_VSERVER"
            Remove-NcFcpService -VserverContext $DESTINATION_VSERVER
-Confirm:$false
            logMessage -message "FC protocol removed on vServer :
$DESTINATION_VSERVER" -type "SUCCESS"
        } else {
            logMessage -message "FCP service is disabled on vServer
$DESTINATION_VSERVER" -type "SUCCESS"
        }

    } catch {
        handleError -errorMessage $_.Exception.Message
    }
}

function disableSvmDataLifs {
    try {
        logMessage -message "Finding all data lifs on vServer :
$DESTINATION_VSERVER"
        $dataLifs = Get-NcNetInterface -Vserver $DESTINATION_VSERVER |
Where-Object { $_.Role -contains "data_core" }
        $dataLifs | Select-Object -Property InterfaceName, OpStatus,
DataProtocols, Vserver, Address

        logMessage -message "Disabling all data lifs on vServer :
$DESTINATION_VSERVER"
        # Disable the filtered data LIFs
        foreach ($lif in $dataLifs) {
            $disableLif = Set-NcNetInterface -Vserver $DESTINATION_VSERVER
-Name $lif.InterfaceName -AdministrativeStatus down -ErrorAction Stop
            $disableLif | Select-Object -Property InterfaceName, OpStatus,
DataProtocols, Vserver, Address
        }
        logMessage -message "Disabled all data lifs on vServer :
$DESTINATION_VSERVER" -type "SUCCESS"
```

```
    } catch {
        handleError -errorMessage $_.Exception.Message
    }
}

function configureMultiAdminApproval {
    try {

        # check if multi admin verification is enabled
        logMessage -message "Checking if multi-admin verification is
enabled"
        $maaConfig = Invoke-NcSsh -Name $DESTINATION_ONTAP_CLUSTER_MGMT_IP
-Credential $DESTINATION_ONTAP_CREDS -Command "set -privilege advanced;
security multi-admin-verify show"
        if ($maaConfig.Value -match "Enabled" -and $maaConfig.Value -match
"true") {
            $maaConfig
            logMessage -message "Multi-admin verification is configured
and enabled" -type "SUCCESS"
        } else {
            logMessage -message "Setting Multi-admin verification rules"
            # Define the commands to be restricted
            $rules = @(
                "cluster peer delete",
                "vserver peer delete",
                "volume snapshot policy modify",
                "volume snapshot rename",
                "vserver audit modify",
                "vserver audit delete",
                "vserver audit disable"
            )
            foreach($rule in $rules) {
                Invoke-NcSsh -Name $DESTINATION_ONTAP_CLUSTER_MGMT_IP
-Credential $DESTINATION_ONTAP_CREDS -Command "security multi-admin-verify
rule create -operation `"$rule`""
            }

            logMessage -message "Creating multi admin verification group
for ONTAP Cluster $DESTINATION_ONTAP_CLUSTER_MGMT_IP, Group name :
$MULTI_ADMIN_APPROVAL_GROUP_NAME, Users : $MULTI_ADMIN_APPROVAL_USERS,
Email : $MULTI_ADMIN_APPROVAL_EMAIL"
            Invoke-NcSsh -Name $DESTINATION_ONTAP_CLUSTER_MGMT_IP
-Credential $DESTINATION_ONTAP_CREDS -Command "security multi-admin-verify
approval-group create -name $MULTI_ADMIN_APPROVAL_GROUP_NAME -approvers
$MULTI_ADMIN_APPROVAL_USERS -email `"$MULTI_ADMIN_APPROVAL_EMAIL`""
            logMessage -message "Created multi admin verification group
```

```
for ONTAP Cluster $DESTINATION_ONTAP_CLUSTER_MGMT_IP, Group name :
$MULTI_ADMIN_APPROVAL_GROUP_NAME, Users : $MULTI_ADMIN_APPROVAL_USERS,
Email : $MULTI_ADMIN_APPROVAL_EMAIL" -type "SUCCESS"

            logMessage -message "Enabling multi admin verification group
$MULTI_ADMIN_APPROVAL_GROUP_NAME"
            Invoke-NcSsh -Name $DESTINATION_ONTAP_CLUSTER_MGMT_IP
-Credential $DESTINATION_ONTAP_CREDS -Command "security multi-admin-verify
modify -approval-groups $MULTI_ADMIN_APPROVAL_GROUP_NAME -required
-approvers 1 -enabled true"
            logMessage -message "Enabled multi admin verification group
$MULTI_ADMIN_APPROVAL_GROUP_NAME" -type "SUCCESS"

            logMessage -message "Enabling multi admin verification for
ONTAP Cluster $DESTINATION_ONTAP_CLUSTER_MGMT_IP"
            Invoke-NcSsh -Name $DESTINATION_ONTAP_CLUSTER_MGMT_IP
-Credential $DESTINATION_ONTAP_CREDS -Command "security multi-admin-verify
modify -enabled true"
            logMessage -message "Successfully enabled multi admin
verification for ONTAP Cluster $DESTINATION_ONTAP_CLUSTER_MGMT_IP" -type
"SUCCESS"

            logMessage -message "Enabling multi admin verification for
ONTAP Cluster $DESTINATION_ONTAP_CLUSTER_MGMT_IP"
            Invoke-NcSsh -Name $DESTINATION_ONTAP_CLUSTER_MGMT_IP
-Credential $DESTINATION_ONTAP_CREDS -Command "security multi-admin-verify
modify -enabled true"
            logMessage -message "Successfully enabled multi admin
verification for ONTAP Cluster $DESTINATION_ONTAP_CLUSTER_MGMT_IP" -type
"SUCCESS"
        }

    } catch {
        handleError -errorMessage $_.Exception.Message
    }
}

function additionalSecurityHardening {
    try {
        $command = "set -privilege advanced -confirmations off;security
protocol modify -application telnet -enabled false;"
        logMessage -message "Disabling Telnet"
        Invoke-NcSsh -Name $DESTINATION_ONTAP_CLUSTER_MGMT_IP -Credential
$DESTINATION_ONTAP_CREDS -Command $command
        logMessage -message "Disabled Telnet" -type "SUCCESS"
```

```
        #$command = "set -privilege advanced -confirmations off;security
config modify -interface SSL -is-fips-enabled true;"
        #logMessage -message "Enabling Global FIPS"
        ##Invoke-SSHCommand -SessionId $sshSession.SessionId -Command
$command -ErrorAction Stop
        #logMessage -message "Enabled Global FIPS" -type "SUCCESS"

        $command = "set -privilege advanced -confirmations off;network
interface service-policy modify-service -vserver cluster2 -policy default-
management -service management-https -allowed-addresses $ALLOWED_IPS;"
        logMessage -message "Restricting IP addresses $ALLOWED_IPS for
Cluster management HTTPS"
        Invoke-NcSsh -Name $DESTINATION_ONTAP_CLUSTER_MGMT_IP -Credential
$DESTINATION_ONTAP_CREDS -Command $command
        logMessage -message "Successfully restricted IP addresses
$ALLOWED_IPS for Cluster management HTTPS" -type "SUCCESS"

        #logMessage -message "Checking if audit logs volume audit_logs
exists"
        #$volume = Get-NcVol -Vserver $DESTINATION_VSERVER -Name
audit_logs -ErrorAction Stop

        #if($volume) {
        #    logMessage -message "Volume audit_logs already exists!
Skipping creation"
        #} else {
        #    # Create audit logs volume
        #    logMessage -message "Creating audit logs volume : audit_logs"
        #    New-NcVol -Name audit_logs -Aggregate
$DESTINATION_AGGREGATE_NAME -Size 5g -ErrorAction Stop | Select-Object
-Property Name, State, TotalSize, Aggregate, Vserver
        #    logMessage -message "Volume audit_logs created successfully"
-type "SUCCESS"
        #}

        ## Mount audit logs volume to path /vol/audit_logs
        #logMessage -message "Creating junction path for volume audit_logs
at path /vol/audit_logs for vServer $DESTINATION_VSERVER"
        #Mount-NcVol -VserverContext $DESTINATION_VSERVER -Name audit_logs
-JunctionPath /audit_logs | Select-Object -Property Name, -JunctionPath
        #logMessage -message "Created junction path for volume audit_logs
at path /vol/audit_logs for vServer $DESTINATION_VSERVER" -type "SUCCESS"

        #logMessage -message "Enabling audit logging for vServer
$DESTINATION_VSERVER at path /vol/audit_logs"
        #$command = "set -privilege advanced -confirmations off;vserver
```

```
audit create -vserver $DESTINATION_VSERVER -destination /audit_logs
-format xml;"
        #Invoke-SSHCommand -SessionI  $sshSession.SessionId -Command
$command -ErrorAction Stop
        #logMessage -message "Successfully enabled audit logging for
vServer $DESTINATION_VSERVER at path /vol/audit_logs"

    } catch {
        handleError -errorMessage $_.Exception.Message
    }
}
```

使用PowerShell验证ONTAP网络存储

强大的网络存储应能够抵御复杂的攻击、即使攻击者拥有凭据来通过提升的Privileges访问环境也是如此。

一旦制定了规则、尝试删除存储端的快照(假设攻击者能够以某种方式进入)将失败。通过施加必要的限制并保护系统、所有强化设置也是如此。

用于按计划验证配置的PowerShell代码示例。

```
function analyze {

    for($i = 0; $i -lt $DESTINATION_VOLUME_NAMES.Length; $i++) {
        try {
            # checking if volume is of type SnapLock Compliance
            logMessage -message "Checking if SnapLock Compliance volume
$($DESTINATION_VOLUME_NAMES[$i]) exists in vServer $DESTINATION_VSERVER"
            $volume = Get-NcVol -Vserver $DESTINATION_VSERVER -Volume
$DESTINATION_VOLUME_NAMES[$i] | Select-Object -Property Name, State,
TotalSize, Aggregate, Vserver, Snaplock | Where-Object { $_.Snaplock.Type
-eq "compliance" }
            if($volume) {
                $volume
                logMessage -message "SnapLock Compliance volume
$($DESTINATION_VOLUME_NAMES[$i]) exists in vServer $DESTINATION_VSERVER"
-type "SUCCESS"
            } else {
                handleError -errorMessage "SnapLock Compliance volume
$($DESTINATION_VOLUME_NAMES[$i]) does not exist in vServer
$DESTINATION_VSERVER. Recommendation: Run the script with SCRIPT_MODE
`"configure`" to create and configure the cyber vault SnapLock Compliance
volume"
            }
```

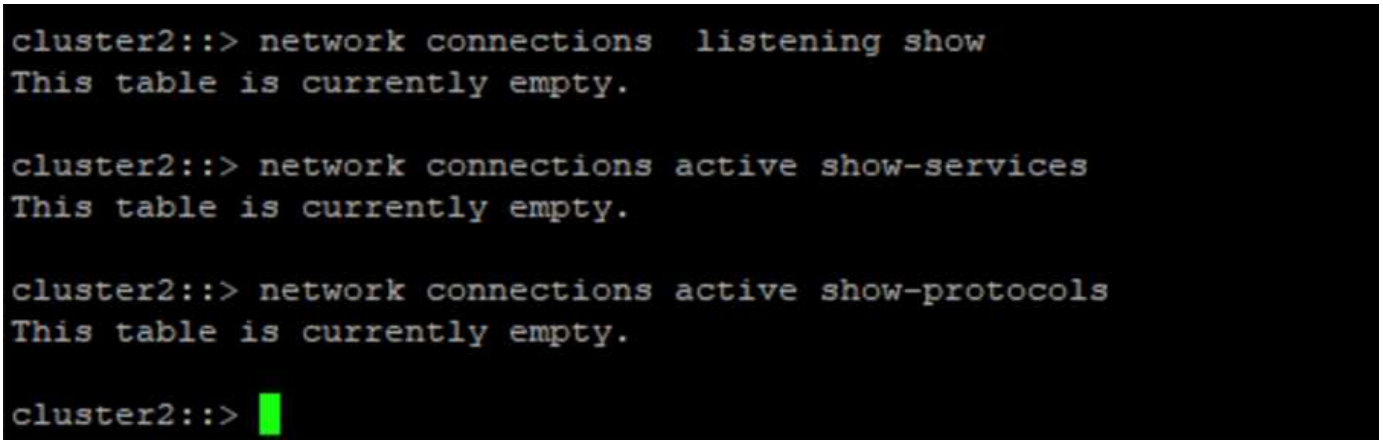
```
            # checking SnapMirror relationship
            logMessage -message "Checking if SnapMirror relationship
exists between source volume $($SOURCE_VOLUME_NAMES[$i]) and destination
SnapLock Compliance volume $($DESTINATION_VOLUME_NAMES[$i])"
            $snapmirror = Get-NcSnapmirror | Select-Object SourceCluster,
SourceLocation, DestinationCluster, DestinationLocation, Status,
MirrorState | Where-Object { $_.SourceCluster -eq
$SOURCE_ONTAP_CLUSTER_NAME -and $_.SourceLocation -eq "$($SOURCE_VSERVER)
:$($SOURCE_VOLUME_NAMES[$i])" -and $_.DestinationCluster -eq
$DESTINATION_ONTAP_CLUSTER_NAME -and $_.DestinationLocation -eq "
$($DESTINATION_VSERVER):$($DESTINATION_VOLUME_NAMES[$i])" -and $_.Status
-eq "snapmirrored" }
            if($snapmirror) {
                $snapmirror
                logMessage -message "SnapMirror relationship successfully
configured and in healthy state" -type "SUCCESS"
            } else {
                handleError -errorMessage "SnapMirror relationship does
not exist between the source volume $($SOURCE_VOLUME_NAMES[$i]) and
destination SnapLock Compliance volume $($DESTINATION_VOLUME_NAMES[$i])
(or) SnapMirror status uninitialized/unhealthy. Recommendation: Run the
script with SCRIPT_MODE `"configure`" to create and configure the cyber
vault SnapLock Compliance volume and configure the SnapMirror
relationship"
            }
        }
        catch {
            handleError -errorMessage $_.Exception.Message
        }
    }

    try {

        # checking NFS service is disabled
        logMessage -message "Checking if NFS service is disabled on
vServer $DESTINATION_VSERVER"
        $nfsService = Get-NcNfsService
        if($nfsService) {
            handleError -errorMessage "NFS service running on vServer
$DESTINATION_VSERVER. Recommendation: Run the script with SCRIPT_MODE
`"configure`" to disable NFS on vServer $DESTINATION_VSERVER"
        } else {
            logMessage -message "NFS service is disabled on vServer
$DESTINATION_VSERVER" -type "SUCCESS"
        }
```

```
        # checking CIFS/SMB server is disabled
        logMessage -message "Checking if CIFS/SMB server is disabled on
vServer $DESTINATION_VSERVER"
        $cifsServer = Get-NcCifsServer
        if($cifsServer) {
            handleError -errorMessage "CIFS/SMB server running on vServer
$DESTINATION_VSERVER. Recommendation: Run the script with SCRIPT_MODE
`"configure`" to disable CIFS/SMB on vServer $DESTINATION_VSERVER"
        } else {
            logMessage -message "CIFS/SMB server is disabled on vServer
$DESTINATION_VSERVER" -type "SUCCESS"
        }

        # checking iSCSI service is disabled
        logMessage -message "Checking if iSCSI service is disabled on
vServer $DESTINATION_VSERVER"
        $iscsiService = Get-NcIscsiService
        if($iscsiService) {
            handleError -errorMessage "iSCSI service running on vServer
$DESTINATION_VSERVER. Recommendation: Run the script with SCRIPT_MODE
`"configure`" to disable iSCSI on vServer $DESTINATION_VSERVER"
        } else {
            logMessage -message "iSCSI service is disabled on vServer
$DESTINATION_VSERVER" -type "SUCCESS"
        }

        # checking FCP service is disabled
        logMessage -message "Checking if FCP service is disabled on
vServer $DESTINATION_VSERVER"
        $fcpService = Get-NcFcpService
        if($fcpService) {
            handleError -errorMessage "FCP service running on vServer
$DESTINATION_VSERVER. Recommendation: Run the script with SCRIPT_MODE
`"configure`" to disable FCP on vServer $DESTINATION_VSERVER"
        } else {
            logMessage -message "FCP service is disabled on vServer
$DESTINATION_VSERVER" -type "SUCCESS"
        }

        # checking if all data lifs are disabled on vServer
        logMessage -message "Finding all data lifs on vServer :
$DESTINATION_VSERVER"
        $dataLifs = Get-NcNetInterface -Vserver $DESTINATION_VSERVER |
Where-Object { $_.Role -contains "data_core" }
        $dataLifs | Select-Object -Property InterfaceName, OpStatus,
```

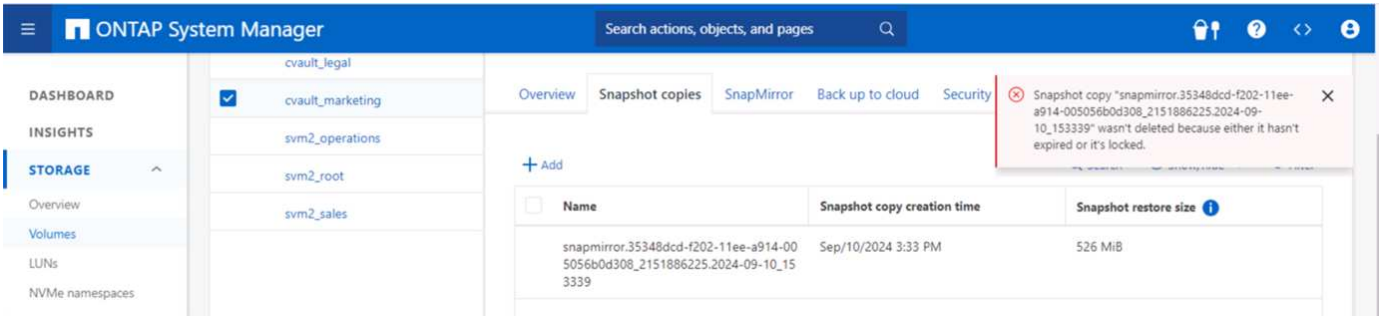
```
DataProtocols, Vserver, Address

        logMessage -message "Checking if all data lifs are disabled for
vServer : $DESTINATION_VSERVER"
        # Disable the filtered data LIFs
        foreach ($lif in $dataLifs) {
            $checkLif = Get-NcNetInterface -Vserver $DESTINATION_VSERVER
-Name $lif.InterfaceName | Where-Object { $_.OpStatus -eq "down" }
            if($checkLif) {
                logMessage -message "Data lif $($lif.InterfaceName)
disabled for vServer $DESTINATION_VSERVER" -type "SUCCESS"
            } else {
                handleError -errorMessage "Data lif $($lif.InterfaceName)
is enabled. Recommendation: Run the script with SCRIPT_MODE `"configure`"
to disable Data lifs for vServer $DESTINATION_VSERVER"
            }
        }
        logMessage -message "All data lifs are disabled for vServer :
$DESTINATION_VSERVER" -type "SUCCESS"

        # check if multi-admin verification is enabled
        logMessage -message "Checking if multi-admin verification is
enabled"
        $maaConfig = Invoke-NcSsh -Name $DESTINATION_ONTAP_CLUSTER_MGMT_IP
-Credential $DESTINATION_ONTAP_CREDS -Command "set -privilege advanced;
security multi-admin-verify show"
        if ($maaConfig.Value -match "Enabled" -and $maaConfig.Value -match
"true") {
            $maaConfig
            logMessage -message "Multi-admin verification is configured
and enabled" -type "SUCCESS"
        } else {
            handleError -errorMessage "Multi-admin verification is not
configured or not enabled. Recommendation: Run the script with SCRIPT_MODE
`"configure`" to enable and configure Multi-admin verification"
        }

        # check if telnet is disabled
        logMessage -message "Checking if telnet is disabled"
        $telnetConfig = Invoke-NcSsh -Name
$DESTINATION_ONTAP_CLUSTER_MGMT_IP -Credential $DESTINATION_ONTAP_CREDS
-Command "set -privilege advanced; security protocol show -application
telnet"
        if ($telnetConfig.Value -match "enabled" -and $telnetConfig.Value
-match "false") {
            logMessage -message "Telnet is disabled" -type "SUCCESS"
```

```
        } else {
            handleError -errorMessage "Telnet is enabled. Recommendation:
Run the script with SCRIPT_MODE `"configure`" to disable telnet"
        }

        # check if network https is restricted to allowed IP addresses
        logMessage -message "Checking if HTTPS is restricted to allowed IP
addresses $ALLOWED_IPS"
        $networkServicePolicy = Invoke-NcSsh -Name
$DESTINATION_ONTAP_CLUSTER_MGMT_IP -Credential $DESTINATION_ONTAP_CREDS
-Command "set -privilege advanced; network interface service-policy show"
        if ($networkServicePolicy.Value -match "management-https:
$($ALLOWED_IPS)") {
            logMessage -message "HTTPS is restricted to allowed IP
addresses $ALLOWED_IPS" -type "SUCCESS"
        } else {
            handleError -errorMessage "HTTPS is not restricted to allowed
IP addresses $ALLOWED_IPS. Recommendation: Run the script with SCRIPT_MODE
`"configure`" to restrict allowed IP addresses for HTTPS management"
        }
    }
    catch {
        handleError -errorMessage $_.Exception.Message
    }
}
```

此屏幕截图显示存储控制器上没有连接。



此屏幕截图显示无法篡改快照。



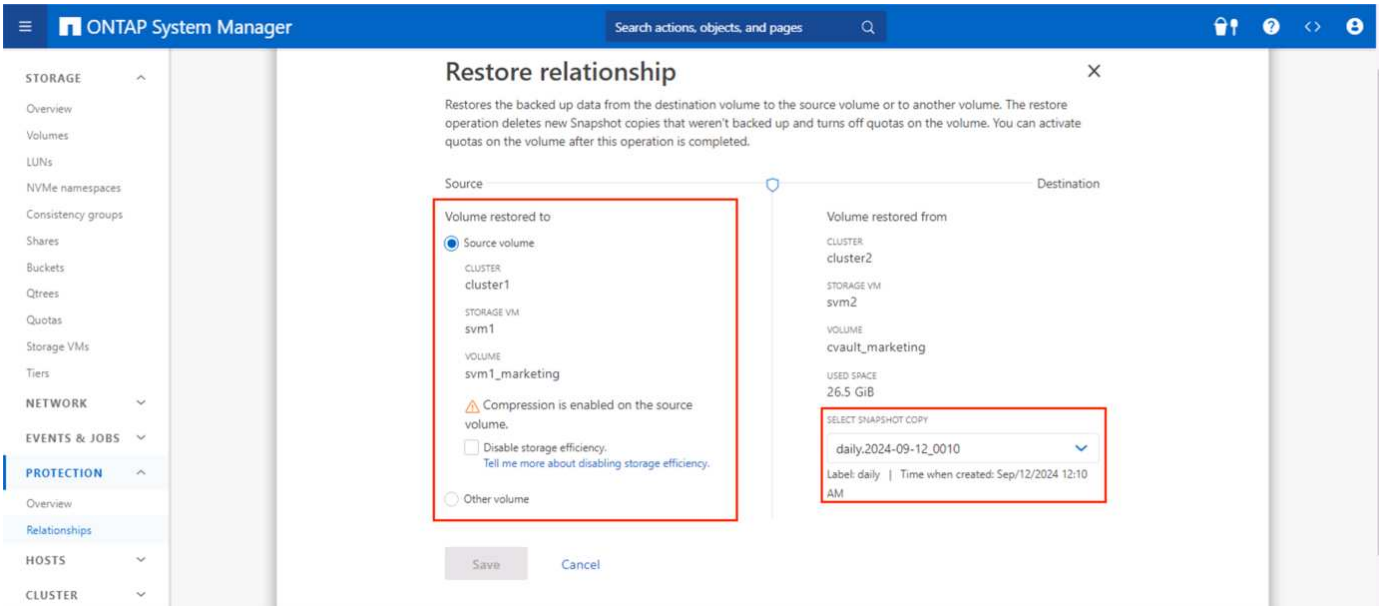
要验证并确认气隙功能、请执行以下步骤：

- 测试网络隔离功能、以及在未传输数据时将连接置于静音状态的功能。
- 验证是否无法从允许的IP地址以外的任何实体访问管理接口。
- 验证是否已实施多管理员验证、以提供额外的审批层。
- 验证是否能够通过命令行界面和REST API进行访问
- 从源中、触发传输操作以存储、并确保存储的副本无法修改。
- 尝试删除传输到存储的不可更改的Snapshot副本。
- 尝试通过篡改系统时钟来修改保留期限。

ONTAP网络存储数据恢复

如果生产数据中心中的数据被销毁、则网络存储中的数据可以安全地恢复到所选环境中。与物理隔离解决方案不同、隔离ONTAP网络存储是使用SnapLock Compliance和SnapMirror等本机ONTAP功能构建的。结果是恢复过程既快速又易于执行。

如果发生勒索软件攻击并需要从网络存储中恢复、则恢复过程非常简单、因为网络存储中的Snapshot副本用于恢复加密数据。



如果需要提供更快的方法、以便在必要时将数据恢复联机、从而快速验证、隔离和分析数据以进行恢复。通过

与FlexClone结合使用并将SnapLock类型选项设置为非SnapLock类型、可以轻松实现这一点。

> 从SnapVault.13.1开始、可以通过创建FlexClone并将SnapLock类型选项设置为"NON-SnapVault"来即时还原SnapLock SnapLock存储关系的目标SnapLock卷上锁定的ONTAP 9副本。执行卷克隆创建操作时、请将Snapshot副本指定为"parent快照"。有关创建SnapLock类型的FlexClone卷的详细信息"此处。"

> 从网络存储执行恢复过程将确保制定正确的步骤、以便连接到网络存储并检索数据。规划和测试该过程对于网络攻击事件期间的任何恢复都至关重要。

其他注意事项

在设计和部署基于ONTAP的网络存储时、还需要考虑一些其他注意事项。

容量规模估算注意事项

ONTAP网络存储目标卷所需的磁盘空间量取决于多种因素、其中最重要的因素是源卷中数据的更改率。目标卷上的备份计划和Snapshot计划都会影响目标卷上的磁盘使用量、源卷上的更改率不太可能保持不变。最好提供一个额外存储容量缓冲区、以满足将来最终用户或应用程序行为的变化所需的容量。

要估算ONTAP中1个月保留关系的规模、需要根据多个因素计算存储需求、包括主数据集的大小、数据更改率(每日更改率)以及重复数据删除和数据压缩节省的空间(如果适用)。

下面是分步方法：

第一步是了解您使用网络存储保护的源卷的大小。这是最初将复制到网络存储目标的基本数据量。接下来、估算数据集的每日变更率。这是每天更改的数据百分比。充分了解数据的动态性至关重要。

例如：

- 主数据集大小= 5 TB
- 每日变更率= 5%(0.05)
- 重复数据删除和数据压缩效率= 50%(0.50)

现在、让我们来了解一下计算结果：

- 计算每日数据变更率：

```
Changed data per day = 5000 * 5% = 250GB
```

- 计算30天内的总更改数据：

```
Total changed data in 30 days = 250 GB * 14 = 3.5TB
```

- 计算所需的总存储：

```
TOTAL = 5TB + 3.5TB = 8.5TB
```

- 应用重复数据删除和数据压缩节省量：

```
EFFECTIVE = 8.5TB * 50% = 4.25TB
```

存储需求摘要

- 如果没有效率：存储30天的网络存储数据需要* 8.5 TB*。
- 效率为50%：在执行重复数据删除和数据压缩后，需要*4.25TB*的存储。

> 由于元数据的原因、Snapshot副本可能会产生额外开销、但这种开销通常很小。

> 如果每天创建多个备份、请按每天创建的Snapshot副本数量调整计算结果。

> 请考虑数据随时间的增长、以确保规模估算适应未来需求。

对主/源的性能影响

由于数据传输是一种拉取操作、因此对主存储性能的影响可能会因工作负载、数据卷和备份频率而异。但是、对主系统的整体性能影响一般是中等的且可管理、因为数据传输旨在将数据保护和备份任务卸载到网络存储系统。在初始关系设置和首次完整备份期间、会将大量数据从主系统传输到网络存储系统(SnapLock Compliance卷)。这可能会导致主系统上的网络流量和I/O负载增加。初始完整备份完成后、ONTAP只需要跟踪和传输自上次备份以来发生更改的块。这样、与初始复制相比、I/O负载会小得多。增量更新非常高效、对主存储性能的影响微乎其微。存储进程在后台运行、这可减少干扰主系统生产工作负载的机会。

- 确保存储系统具有足够的资源(CPU、内存和IOPS)来处理额外的负载可缓解性能影响。

配置、分析、cron脚本

NetApp创建了一个可下载的脚本、用于配置、验证和计划网络存储关系。

此脚本的作用

- 集群对等
- SVM 对等
- DP卷创建
- SnapMirror关系和初始化
- 对用于网络存储的ONTAP系统进行了加密
- 根据传输计划暂停和恢复关系
- 定期验证安全设置并生成报告以显示任何异常情况

如何使用此脚本

下载脚本并使用脚本、只需执行以下步骤：

- 以管理员身份启动Windows PowerShell。
- 导航到包含脚本的目录。

- 使用 `.\`语法和所需参数执行该脚本

请确保输入了所有信息。首次运行(配置模式)时、它将要求提供生产和新网络存储系统的凭据。之后、它将在系统之间创建SVM对等关系(如果不存在)、卷和SnapMirror并对其进行初始化。

cron模式可用于计划暂停和恢复数据传输。

操作模式

自动化脚本提供了3种执行模式- `configure`、 `analyze`和 `cron`。

```
if($SCRIPT_MODE -eq "configure") {
        configure
    } elseif ($SCRIPT_MODE -eq "analyze") {
        analyze
    } elseif ($SCRIPT_MODE -eq "cron") {
        runCron
    }
```

- 配置-执行验证检查并将系统配置为气隙。
- 分析-自动监控和报告功能、用于向监控组发送异常和可疑活动的信息、以确保配置不会偏离。
- cron—要启用已断开连接的基础架构、cron模式会自动禁用LIF并使传输关系处于静机状态。

传输这些选定卷中的数据需要一些时间、具体取决于系统性能和数据量。

```
./script.ps1 -SOURCE_ONTAP_CLUSTER_MGMT_IP "172.21.166.157"
-SOURCE_ONTAP_CLUSTER_NAME "NTAP915_Src" -SOURCE_VSERVER "svm_NFS"
-SOURCE_VOLUME_NAME "Src_RP_Vol01" -DESTINATION_ONTAP_CLUSTER_MGMT_IP
"172.21.166.159" -DESTINATION_ONTAP_CLUSTER_NAME "NTAP915_Destn"
-DESTINATION_VSERVER "svm_nim_nfs" -DESTINATION_AGGREGATE_NAME
"NTAP915_Destn_01_VM_DISK_1" -DESTINATION_VOLUME_NAME "Dst_RP_Vol01_Vault"
-DESTINATION_VOLUME_SIZE "5g" -SNAPLOCK_MIN_RETENTION "15minutes"
-SNAPLOCK_MAX_RETENTION "30minutes" -SNAPMIRROR_PROTECTION_POLICY
"XDPDefault" -SNAPMIRROR_SCHEDULE "5min" -DESTINATION_CLUSTER_USERNAME
"admin" -DESTINATION_CLUSTER_PASSWORD "PASSWORD123"
```

ONTAP网络存储PowerShell解决方案总结

通过利用ONTAP提供的强大的强化方法进行气隙调整、NetApp使您能够创建一个安全、隔离的存储环境、以抵御不断变化的网络威胁。所有这些都是在保持现有存储基础架构的灵活性和效率的同时完成的。这种安全访问使公司能够实现严格的安全和正常运行时间目标、同时最大限度地减少对现有人员、流程和技术框架的更改。

ONTAP网络存储使用ONTAP中的本机功能、这是一种可提供额外保护的简单方法、可为您的数据创建不可变和不可删除的副本。将NetApp基于ONTAP的网络存储添加到整体安全防护中将：

- 创建一个与生产网络和备份网络分离并断开连接的环境、并限制用户对该环境的访问。