PowerShellを使用したONTAPサイバーボールトの作成、強化、検証

NetApp Solutions

NetApp
September 26, 2024

目次

PowerShellを使用したONTAPサイバーボールトの作成、強化、検証

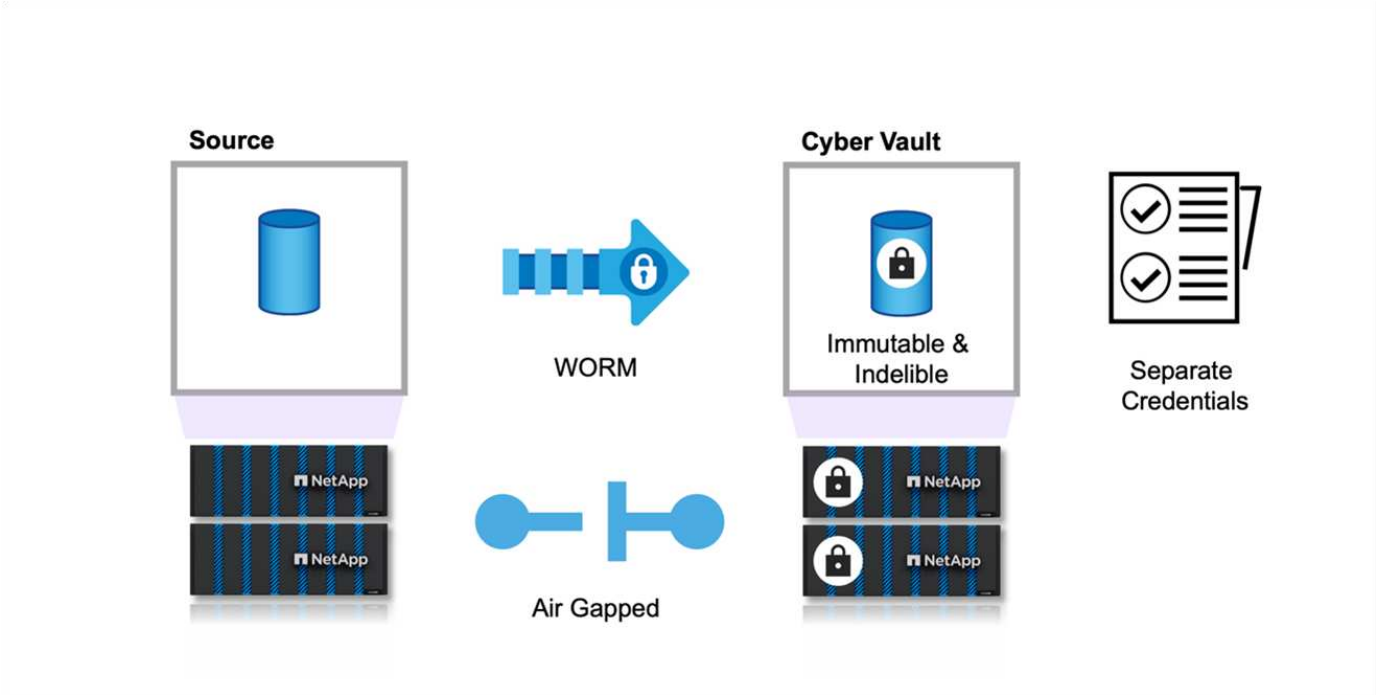
PowerShellを使用したONTAPサイバーボールトの概要

今日のデジタル環境では、組織の重要なデータ資産の保護は単なるベストプラクティスではなく、ビジネス上の必須課題です。サイバー脅威は前例のないペースで進化しており、従来のデータ保護対策では機密情報のセキュリティを確保することはできません。そこで、サイバーボールトが登場します。NetAppの最先端のONTAPベースのソリューションは、高度なエアギャップ技術と堅牢なデータ保護対策を組み合わせて、サイバー脅威に対する断固とした障壁を作り出します。安全なセキュリティ強化テクノロジで最も重要なデータを分離することで、サイバーボールトは攻撃対象を最小限に抑え、最も重要なデータを機密性の高い状態に保ち、必要なときにすぐに利用できるようにします。

サイバーヴォールトは、ファイアウォール、ネットワーク、ストレージなどの複数の保護レイヤで構成されるセキュアなストレージ施設です。これらのコンポーネントは、重要なビジネスオペレーションに必要な重要なリカバリデータを保護します。サイバーボールトのコンポーネントは、バックアップポリシーに基づいて重要な本番データと定期的に同期されますが、それ以外の場合はアクセスできません。この分離された切断されたセットアップにより、サイバー攻撃が本番環境を侵害した場合に、サイバーボールトから信頼性の高い最終的なリカバリを簡単に実行できます。

NetAppを使用すると、ネットワークの設定、LIFの無効化、ファイアウォールルールの更新、外部ネットワークやインターネットからのシステムの分離によって、サイバーヴォールト用のエアギャップを簡単に作成できます。この堅牢なアプローチは、外部ネットワークやインターネットからシステムを効果的に切断し、リモートサイバー攻撃や不正アクセスの試みに対する比類のない保護を提供し、システムをネットワークベースの脅威や侵入から保護します。

これをSnapLock Compliance保護と組み合わせることで、ONTAP管理者やNetAppサポートがデータを変更または削除することはできません。SnapLockは、SECおよびFINRAの規制に照らして定期的に監査され、データの耐障害性が銀行業界のこうした厳しいWORMおよびデータ保持の規制を確実に満たしていることを確認しています。NetAppは、最高機密データを保存するためにNSA CSfCによって検証された唯一のエンタープライズストレージです。



このドキュメントでは、オンプレミスのONTAPストレージ用のNetAppのサイバーヴォールトを、変更不可のSnapshotを使用して別の指定されたONTAPストレージに自動設定する方法について説明します。これにより、増大するサイバー攻撃からの保護が強化され、迅速なリカバリが可能になります。このアーキテクチャの一部として、ONTAPのベストプラクティスに従って構成全体が適用されます。最後のセクションでは、攻撃が発生した場合にリカバリを実行する手順について説明します。

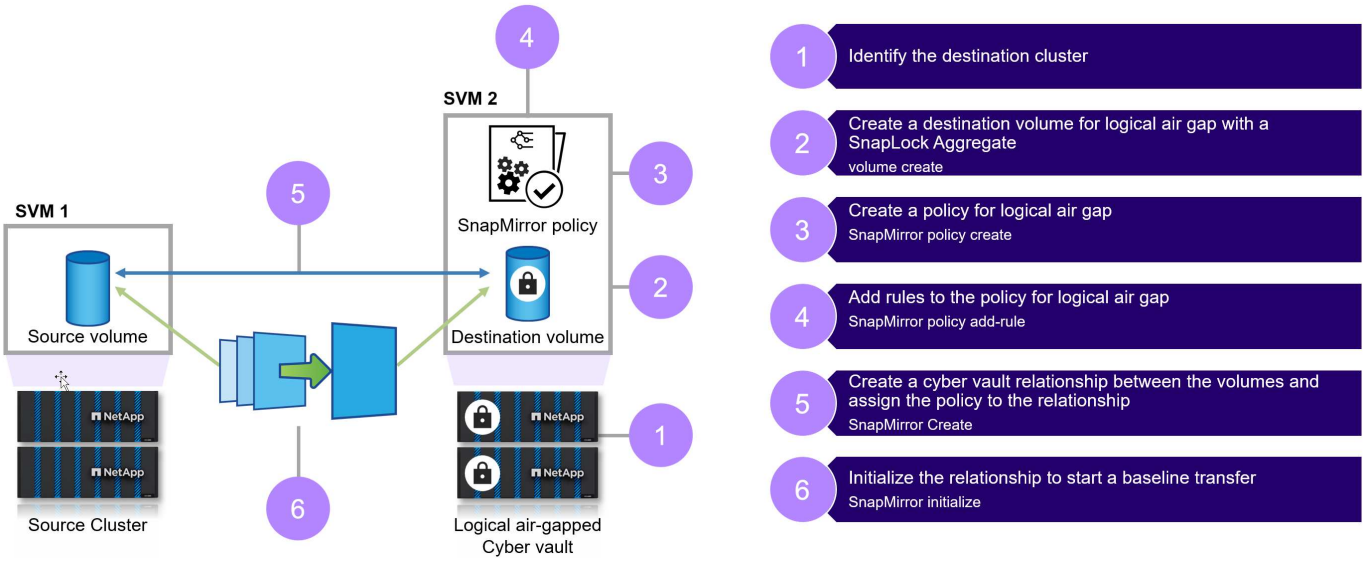
FSx for ONTAPを使用してAWSに指定されたサイバーボールトを作成する場合も、同じソリューションを使用できます。

ONTAPサイバーボールトを作成するための手順の概要

- ピア関係の作成
  - ONTAPストレージを使用する本番サイトは、指定されたサイバーヴォールトONTAPストレージとピアリング
- SnapLock Complianceボリュームの作成
- ラベルを設定するSnapMirror関係とルールを設定する
  - SnapMirror関係と適切なスケジュールが設定されている
- SnapMirror（ヴォールト）転送を開始する前に保持を設定
  - コピーされたデータに保持ロックが適用されるため、内部関係者やデータ障害によるデータの損失を防ぐことができます。これを使用すると、保持期間が切れる前にデータを削除できなくなります。
  - 組織は要件に応じて、このデータを数週間または数カ月保存できます。
- ラベルに基づいてSnapMirror関係を初期化する
  - 初回シードと永久増分転送はSnapMirrorスケジュールに基づいて実行
  - SnapLock Complianceでデータを保護（変更不可、消去不可）し、そのデータをリカバリに利用可能
- 厳格なデータ転送管理を実装

- サイバーボールトは、本番サイトのデータを使用して一定期間ロック解除され、ボールト内のデータと同期されます。転送が完了すると、接続が切断され、閉じられ、再度ロックされます。

- クイックリカバリ
  - プライマリが本番サイトで影響を受ける場合、サイバーボールトのデータは元の本番環境または選択した別の環境に安全にリカバリされます。



解決策コンポーネント

ソースクラスタとデスティネーションクラスタで9.15.1を実行しているNetApp ONTAP。

ONTAP One：NetApp ONTAPのオールインワンライセンス。

ONTAP Oneライセンスで使用される機能：

- SnapLock Compliance
- SnapMirror
- マルチ管理者認証
- ONTAPが公開するすべての強化機能
- サイバーヴォールト用の個別のRBACクレデンシャル

> All ONTAPユニファイド物理アレイはサイバーバックアップにも使用できますが、AFF Cシリーズの容量ベースのフラッシュシステムとFASハイブリッドフラッシュシステムは、この目的のための最も対費用効果の高いプラットフォームです。"ONTAPサイバーヴォールトのサイジング"サイジングのガイダンスについては、を参照してください。

PowerShellを使用したONTAPサイバーヴォールトの作成

従来の方法を使用するエアギャップバックアップでは、スペースを確保し、プライマリメディアとセカンダリメディアを物理的に分離します。メディアをオフサイトに移動したり、接続を切断したりすることで、攻撃者はデータにアクセスできなくなります。こ

れによりデータが保護されますが、リカバリ時間が長くなる可能性があります。SnapLock Complianceでは、物理的な分離は必要ありません。SnapLock Complianceは、保存されたSnapshotのポイントインタイムの読み取り専用コピーを保護します。その結果、データに迅速にアクセスし、削除や消去を防止し、変更や改ざんを防止できます。

前提条件

このドキュメントの次のセクションで説明する手順を開始する前に、次の前提条件を満たしていることを確認してください。

- ソースクラスタでONTAP 9以降が実行されている必要があります。
- ソースアグリゲートとデスティネーションアグリゲートは64ビットである必要があります。
- ソースクラスタとデスティネーションクラスタのピア関係が確立されている必要があります。
- ソースとデスティネーションのSVMのピア関係が確立されている必要があります。
- クラスタピアリングの暗号化が有効になっていることを確認

ONTAPサイバーボールトへのデータ転送を設定するには、いくつかの手順が必要です。プライマリボリュームで、適切なスケジュールを使用して作成するコピーと作成するタイミングを指定するSnapshotポリシーを設定し、ラベルを割り当ててSnapVaultで転送するコピーを指定します。セカンダリでは、転送するSnapshotコピーのラベルと、それらのコピーのうち何個をサイバーバックアップに保持するかを指定するSnapMirrorポリシーを作成する必要があります。これらのポリシーを設定したら、SnapVault関係を作成し、転送スケジュールを確立します。

本ドキュメントでは、プライマリストレージと指定されたONTAPサイバーボールトがすでにセットアップおよび設定されていることを前提としています。

サイバーボールトクラスタは、ソースデータと同じデータセンターにも別のデータセンターにも配置できます。

ONTAPサイバーボールトを作成する手順

1. ONTAP CLIまたはSystem Managerを使用して、コンプライアンスクロックを初期化してください。
2. SnapLock Complianceを有効にしてデータ保護ボリュームを作成します。
3. SnapMirror createコマンドを使用して、SnapVaultデータ保護関係を作成します。
4. デスティネーションボリュームのデフォルトのSnapLock Compliance保持期間を設定します。

デフォルトの保持期間は「最小に設定」です。バックアップデスティネーションであるSnapLockには、デフォルトの保持期間が割り当てられています。この期間の最初の値は、SnapLock Complianceボリュームでは最小0年、最大30年に設定されています。各NetApp Snapshotコピーは、最初はこのデフォルトの保持期間でコミットされます。保持期間は必要に応じてあとから延長できますが、短縮することはできません。

上記には手動の手順が含まれています。セキュリティの専門家は、プロセスを自動化して、エラーの大きなマージンをもたらす手動管理を回避することをお勧めします。以下は、SnapLock Complianceの前提条件と設定、およびクロックの初期化を完全に自動化するコードスニペットです。

次に、ONTAPコンプライアンスクロックを初期化するためのPowerShellコード例を示します。

```
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
    }
}
```

ここでは、ONTAPサイバーボールトを構成するためのPowerShellコード例を示します。

```
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. 上記の手順が完了すると、SnapLock ComplianceとSnapVaultを使用したエアギャップサイバーヴォールトの準備が整います。

Snapshotデータをサイバーバックアップに転送する前に、SnapVault関係を初期化する必要があります。ただし、その前に、ヴォールトを保護するためにセキュリティ強化を実行する必要があります。

PowerShellを使用したONTAPサイバーバックアップの強化

ONTAPサイバーボールトは、従来のソリューションに比べてサイバー攻撃に対する耐障害性が向上します。セキュリティを強化するアーキテクチャを設計する際には、攻撃の表面積を削減するための対策を検討することが重要です。これは、強化されたパスワードポリシーの実装、RBACの有効化、デフォルトのユーザーアカウントのロック、ファイアウォールの設定、ボルトシステムへの変更に対する承認フローの利用など、さまざまな方法で実現できます。さらに、ネットワークアクセスプロトコルを特定のIPアドレスから制限することは、潜在的な脆弱性を制限するのに役立ちます。

ONTAPには、ONTAPストレージを強化するための一連の制御機能が用意されています。を使用して"ONTAPのガイダンスと構成設定"、組織が情報システムの機密性、整合性、可用性に関する規定のセキュリティ目標を達成できるようにします。

強化のベストプラクティス

手動手順

1. 事前定義されたカスタム管理ロールを持つ指定ユーザーを作成します。
2. 新しいIPspaceを作成してネットワークトラフィックを分離します。
3. 新しいIPspaceに新しいSVMを作成します。

4. ファイアウォールルーティングポリシーが適切に設定されていること、およびすべてのルールが定期的に監査され、必要に応じて更新されていることを確認します。

ONTAP CLIまたは自動スクリプトを使用

1. マルチ管理者検証（MFA）による管理の保護
2. クラスタ間の「転送中」の標準データの暗号化を有効にします。
3. 強力な暗号化暗号を使用してSSHを保護し、セキュアなパスワードを適用します。
4. グローバルFIPSを有効にします。
5. TelnetとRemote Shell（RSH；リモートシェル）は無効にする必要があります。
6. デフォルトの管理者アカウントをロックします。
7. データLIFを無効にし、リモートアクセスポイントを保護します。
8. 使用していないプロトコルや無関係なプロトコルやサービスを無効にして削除します。
9. ネットワークトラフィックを暗号化します。
10. スーパーユーザロールと管理ロールを設定する場合は、最小権限の原則を使用します。
11. [Allowed IP]オプションを使用して、特定のIPアドレスからのHTTPSとSSHを制限します。
12. 転送スケジュールに基づいてレプリケーションを休止および再開します。

1～4の箇条書きには、分離されたネットワークの指定やIPspaceの分離など、手動の操作が必要です。この操作は事前に行う必要があります。セキュリティ強化を設定するための詳細については、を参照し"ONTAPセキュリティ強化ガイド"てください。残りは簡単に自動化でき、導入と監視が容易になります。このオーケストレーションされたアプローチの目的は、ヴォールトコントローラの将来を保証するための強化手順を自動化するメカニズムを提供することです。サイバーボールトのエアギャップが開いている期間は、できるだけ短くなっています。SnapVaultはインクリメンタルフォーエバーテクノロジーを活用しています。これは、前回の更新以降の変更のみをサイバーボールトに移動するため、サイバーボールトを開いておく必要がある時間を最小限に抑えることができます。ワークフローをさらに最適化するために、サイバーボールトのオープンはレプリケーションスケジュールに合わせて調整され、接続ウィンドウが最小になります。

ここでは、ONTAPコントローラを強化するためのPowerShellコード例を示します。

```
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
    }
}
```

PowerShellを使用したONTAPサイバーヴォールト検証

堅牢なサイバーボールトは、攻撃者が昇格されたPrivilegesを使用して環境にアクセスするための資格情報を持っている場合でも、高度な攻撃に耐えることができるはずです。

ルールが設定されると、（攻撃者が何らかの方法で侵入できたと仮定して）ボールト側のスナップショットを削除しようとすると失敗します。必要な制限を適用し、システムを保護することで、すべての強化設定にも同じことが当てはまります。

スケジュールに基づいて構成を検証するPowerShellコード例。

```
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"
            }
```

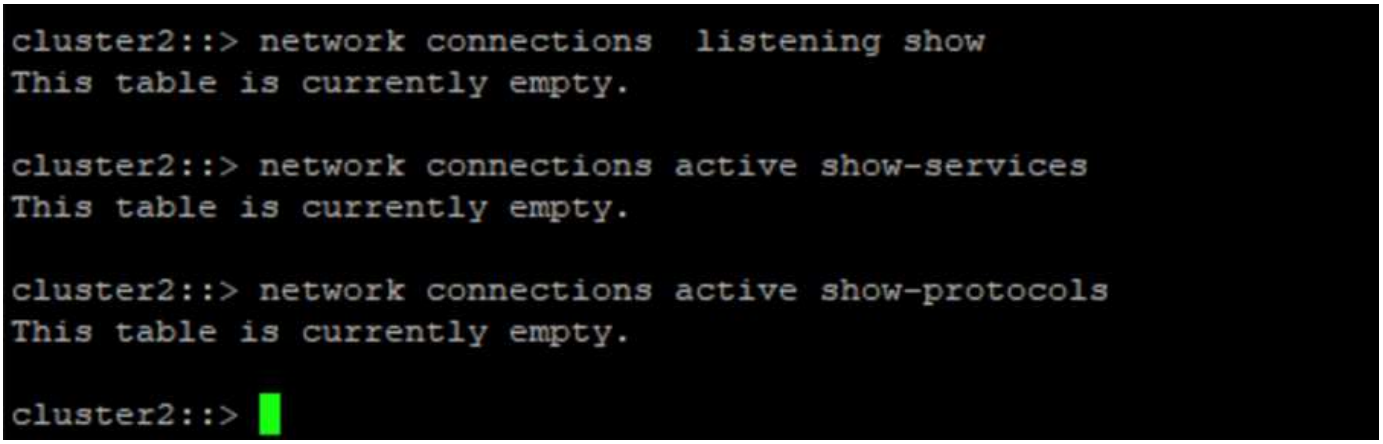
```
            # checking SnapMirror relationship
            logMessage -message "Checking if SnapMirror relationship
exists between source volume $($SOURCE_VOLUME_NAMES[$i]) and destination
SnapLock Compliance volume $($DESTINATION_VOLUME_NAMES[$i])"
            $snapmirror = Get-NcSnapmirror | Select-Object 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,
```

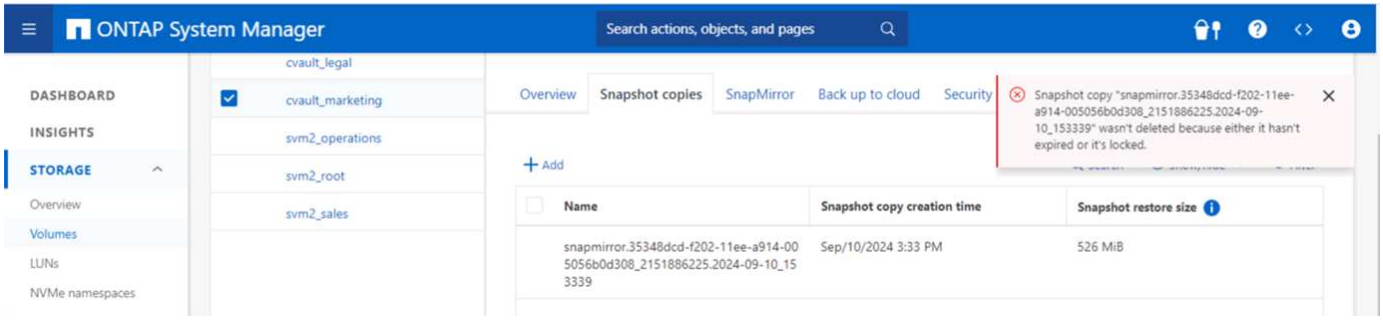
```
DataProtocols, Vserver, Address

        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
    }
}
```

このスクリーンショットは、ボールトコントローラに接続がないことを示しています。



このスクリーンショットは、スナップショットを改ざんする機能がないことを示しています。



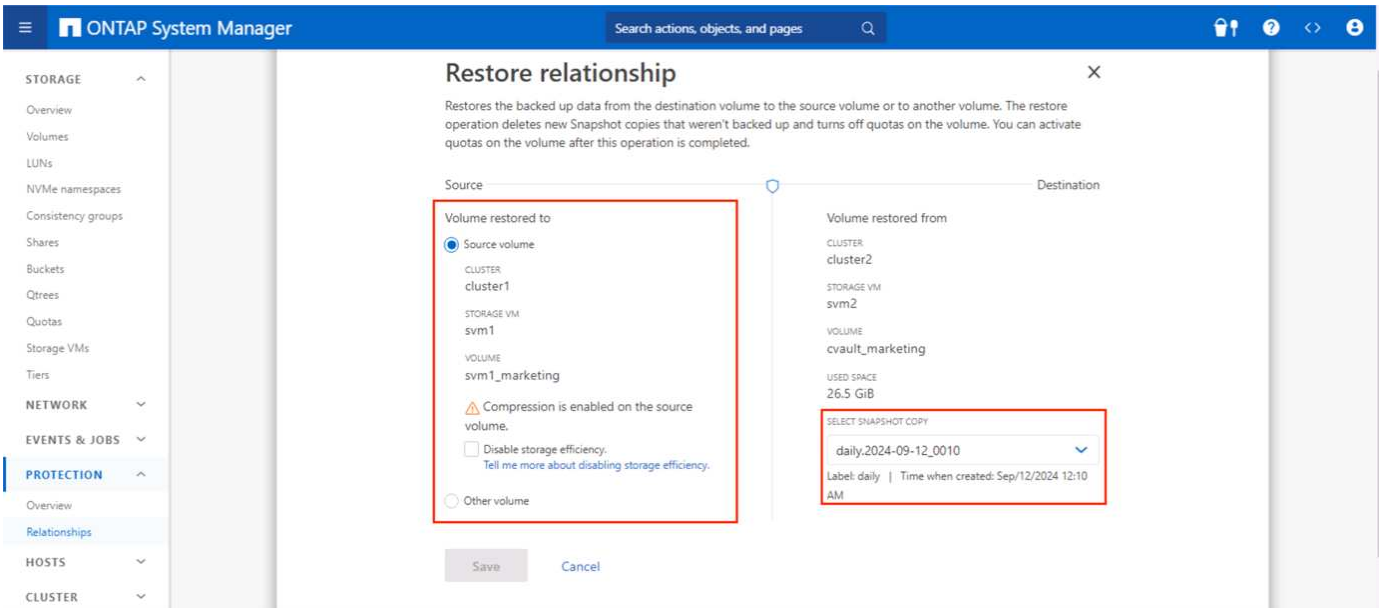
エアギャップ機能を検証して確認するには、次の手順に従います。

- ネットワーク分離機能をテストし、データが転送されていないときに接続を休止する機能をテストします。
- 許可されているIPアドレス以外のエンティティから管理インターフェイスにアクセスできないことを確認してください。
- マルチ管理者認証が設定されていることを確認して、追加の承認層を提供します。
- CLIとREST APIを使用してアクセスできるかどうかを検証
- ソースからヴォールトへの転送処理を開始し、保存されたコピーが変更されないようにします。
- ヴォールトに転送された変更不可のSnapshotコピーを削除してください。
- システムクロックを改ざんして保持期間を変更してください。

ONTAPサイバーヴォールトデータリカバリ

実稼働中のデータセンターでデータが破壊された場合、サイバーボールトのデータを選択した環境に安全にリカバリできます。物理的にエアギャップされたソリューションとは異なり、エアギャップされたONTAPサイバーボールトは、SnapLock ComplianceやSnapMirrorなどのネイティブONTAP機能を使用して構築されています。その結果、迅速かつ簡単に実行できるリカバリプロセスが実現します。

ランサムウェア攻撃が発生し、サイバーボールトからのリカバリが必要になった場合、サイバーボールトに格納されているSnapshotコピーを使用して暗号化されたデータをリストアするため、リカバリプロセスはシンプルで簡単です。



リカバリのためにデータを迅速に検証、分離、分析するために、必要に応じて迅速にデータをオンラインに戻す方法を提供する必要がある場合。これは、FlexCloneでSnapLockタイプオプションを非SnapLockタイプに設定して使用することで簡単に実現できます。

Snapshot.13.1以降では、ロックされたSnapshotコピーをSnapLockバックアップ関係のデスティネーションSnapLock SnapLockボリュームにリストアするには、snapshot-typeオプションを「SnapLock」に設定してONTAP 9 FlexCloneを作成します。ボリュームクローン作成処理を実行する場合は、Snapshotコピーを「parent-snapshot」として指定します。SnapLockタイプのFlexCloneボリュームの作成に関する詳細情報"はい。"

サイバーボールトからリカバリ手順を実行することで、サイバーボールトに接続してデータを取得するための適切な手順が確立されます。この手順の計画とテストは、サイバー攻撃イベント中のリカバリに不可欠です。

その他の考慮事項

ONTAPベースのサイバーボールトを設計および導入する際には、追加の考慮事項があります。

容量サイジングに関する考慮事項

ONTAPサイバーバックアップデスティネーションボリュームに必要なディスクスペースの量はさまざまな要因によって異なりますが、最も重要な要因はソースボリュームのデータの変更率です。デスティネーションボリュームでのバックアップスケジュールとSnapshotスケジュールはどちらもデスティネーションボリュームのディスク使用量に影響し、ソースボリュームでの変更率は一定ではない可能性があります。エンドユーザやアプリケーションの今後の動作の変化に対応するために必要な容量よりも多くのストレージ容量を確保することを推奨します。

ONTAPで保持期間1カ月の関係をサイジングするには、プライマリデータセットのサイズ、データ変更率（日次変更率）、重複排除と圧縮による削減量（該当する場合）など、いくつかの要因に基づいてストレージ要件を計算する必要があります。

ステップバイステップのアプローチは次のとおりです。

最初のステップは、サイバーヴォールトで保護するソースボリュームのサイズを確認することです。これは、サイバーヴォールトデスティネーションに最初にレプリケートされるデータの基本量です。次に、データセットの1日の変更率を見積もります。日 々 変化するデータの割合です。データの動的性について十分に理解しておくことが重要です。

例：

- プライマリデータセットのサイズ= 5TB
- 1日の変更率= 5%（0.05）
- 重複排除と圧縮による削減率= 50%（0.50）

では、計算を見ていきましょう。

- 日単位のデータ変更率を計算します。

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

- 30日間の変更データの合計を計算します。

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

- 必要な総ストレージ容量を計算します。

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

- 重複排除と圧縮による削減効果を適用します。

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

ストレージニーズの概要

- 効率性がない場合：30日間分のサイバーヴォールトデータを保存するには* 8.5TB *が必要です。
- 50%の効率性で、重複排除と圧縮のあとに* 4.25TB *のストレージが必要になります。

Snapshotコピーではメタデータによってオーバーヘッドが増える可能性がありますが、通常はこれはわずかです。

1日に複数のバックアップが作成される場合は、1日に作成されるSnapshotコピーの数で計算を調整します。

長期的なデータ増加を考慮して、サイジングが将来のニーズにも対応できるようにします。

プライマリ/ソースへのパフォーマンスへの影響

データ転送はプル処理であるため、プライマリストレージのパフォーマンスへの影響は、ワークロード、データボリューム、バックアップの頻度によって異なります。ただし、データ転送はデータ保護とバックアップタスクをサイバーヴォールトストレージシステムにオフロードするように設計されているため、プライマリシス

テム全体のパフォーマンスへの影響は一般に中程度で管理可能です。関係の初期セットアップおよび初回のフルバックアップでは、大量のデータがプライマリシステムからサイバーバックアップシステム（SnapLock Complianceボリューム）に転送されます。これにより、プライマリシステムのネットワークトラフィックやI/O負荷が増加する可能性があります。初回のフルバックアップが完了すると、ONTAPは前回のバックアップ以降に変更されたブロックを追跡して転送するだけで済みます。これにより、最初のレプリケーションに比べてI/O負荷が大幅に軽減されます。差分更新は効率的で、プライマリストレージのパフォーマンスへの影響は最小限です。ボールトプロセスはバックグラウンドで実行されるため、プライマリシステムの本番ワークロードに干渉する可能性が低くなります。

- ストレージシステムに追加の負荷を処理するための十分なリソース（CPU、メモリ、IOPS）があることを確認すると、パフォーマンスへの影響が軽減されます。

設定、分析、cronスクリプト

NetAppでは、ダウンロードしてサイバーバックアップ関係の設定、検証、スケジュール設定に使用できる単一のスクリプトを作成しました。

このスクリプトの機能

- クラスタピアリング
- SVMピアリング
- DPボリュームノサクセイ
- SnapMirror関係と初期化
- サイバーボールトに使用するONTAPシステムを強化
- 転送スケジュールに基づいて関係を休止して再開する
- セキュリティ設定を定期的に検証し、異常を示すレポートを生成

このスクリプトの使用方法

スクリプトをダウンロードして使用するには、次の手順を実行します。

- Windows PowerShellを管理者として起動します。
- スクリプトが格納されているディレクトリに移動します。
- `.\`構文と必要なパラメータを使用してスクリプトを実行します。

すべての情報を入力してください。最初の実行（構成モード）では、本番システムと新しいサイバーボールトシステムの両方のクレデンシャルを要求します。その後、システム間にSVMピア（存在しない場合）、ボリューム、およびSnapMirrorが作成されて初期化されます。

cronモードを使用すると、データ転送の休止と再開をスケジュールできます。

動作モード

自動化スクリプトには `configure`、 `analyze`および3つの実行モードが用意されています `cron`。

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

- Configure -検証チェックを実行し、システムをエアギャップとして設定します。
- 分析-自動化された監視およびレポート機能。監視グループに異常や疑わしいアクティビティに関する情報を送信して、設定がドリフトしないようにします。
- cron -切断されたインフラを有効にするには、cronモードでLIFを自動的に無効にして転送関係を休止します。

システムのパフォーマンスとデータ量の両方に応じて、選択したボリュームのデータの転送には時間がかかります。

```
./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サイバーヴォールトPowerShellソリューションのまとめ

NetAppでは、ONTAPが提供する堅牢なセキュリティ強化手法とエアギャップを活用することで、進化するサイバー脅威に対して耐障害性を備えたセキュアで分離されたストレージ環境を構築できます。これらはすべて、既存のストレージインフラの即応性と効率性を維持しながら実現されます。このセキュアなアクセスにより、企業は既存の人員、プロセス、およびテクノロジーフレームワークへの変更を最小限に抑えながら、厳しい安全性と稼働時間の目標を達成できます。

ONTAPのサイバーヴォールトは、ONTAPのネイティブ機能を使用してデータを簡単に保護し、書き換え不可能なコピーを作成します。NetAppのONTAPベースのサイバーボールトを全体的なセキュリティ体制に追加すると、次のことが可能になります。

- 本番ネットワークとバックアップネットワークに分離され、接続されていない環境を作成し、その環境へのユーザアクセスを制限します。

著作権に関する情報

商標に関する情報