

# From The rchitect

An Veeam Architect Paper

## **Veeam Hardened Repository with RedHat Enterprise Linux**

Author Timothy Dewin

## Contents

|   |    |
|---|----|
| Veeam Hardened Repository with RedHat Enterprise Linux..... | 1  |
| Intro.....  | 4  |
| Executive Summary.....                                      | 4  |
| Objectives.....   | 4  |
| Acronyms.....   | 5  |
| How Veeam Hardened Repositories work.....                   | 6  |
| Paranoia level.....   | 7  |
| Consult all resources.....                                  | 7  |
| Encryption.....   | 7  |
| Command line style.....                                     | 8  |
| Management Considerations.....                              | 11 |
| Hardware vs VM.....   | 11 |
| Remote management.....                                      | 11 |
| Physical access.....  | 11 |
| Install.....  | 12 |
| Hardware setup.....   | 12 |
| Installation type.....                                      | 13 |
| Begin Install.....  | 13 |
| Time Configuration.....                                     | 14 |
| Software selection.....                                     | 14 |
| NIST 800-171 Security Profile.....                          | 14 |
| User setup.....   | 15 |
| After Install.....  | 16 |
| Securing your admin user with key based authentication..... | 16 |
| Initial update.....   | 19 |
| Automatic updates.....                                      | 20 |
| Enabling firewall.....                                      | 20 |
| Adding a user that will act as a data mover.....            | 22 |
| Preparing your backup volumes.....                          | 22 |
| Reboot.....   | 25 |
| Registering the repository.....                             | 26 |

|   |    |
|---|----|
| Adding the repository.....                      | 26 |
| Multiple physical servers.....                  | 31 |
| Post registration hardening.....                | 32 |
| Locking down after registration.....            | 32 |
| Hardening SSH at startup.....                   | 32 |
| Secure the firewall openings after setup.....   | 33 |
| Virtual offline repository.....                 | 33 |
| Hardening Chrony.....                           | 34 |
| Monitoring your Veeam Hardened Repository ..... | 35 |
| Monitoring disk usage .....                     | 35 |
| Monitoring the Veeam Processes.....             | 35 |
| Checking the logs.....                          | 36 |
| Analysing the backup files .....                | 37 |
| Immutability Report.....                        | 39 |

## Intro

### Executive Summary

In the last decade, we have seen a rise in cyber warfare. Today, almost every company in the world, big and small has encountered some form of Ransomware. We have seen that security has become more and more critical for every organization. When environments are breached, backup turns out to be the last line of defense and the first line of recovery. Therefore, it is critical to make sure that it itself does not fall victim to Ransomware

In this document we will discuss how to set up a Veeam Hardened Repository with immutability. These kinds of repositories, when setup correctly, can help you defend a ransomware attack by making sure that your backups will not be encrypted and that you can recover from such attacks successfully.

We used RHEL 8 in this document as it is a popular distribution in Enterprise environments. Most of the items discussed in this document can be applied to other distributions although RHEL has a strong focus on security and compliance. For example, RHEL allows you to install the system under a NIST profile that enforce extra security rules out of the box. A good example here is the use of FAPolicyd, a software framework that blocks applications from running if they are unknown to the system.

### Objectives

It is important to understand that the target audience of this document is a typical Windows admin who might have little experience with Linux, but still wants to setup a hardened repository.

The guide will give an overview of how to get started and make sure you have a secure repository without having to look through multiple resources. If you are an experienced Linux admin, you can still follow the step-by-step process. Most of the configuration is command line so making a complete script out of this should be fairly straight forward.

Using this guide, you should be able to setup a hardened repository. For example, we will go through the install process, also explaining how you format and mount the backup target to the Linux filesystem.

## Acronyms

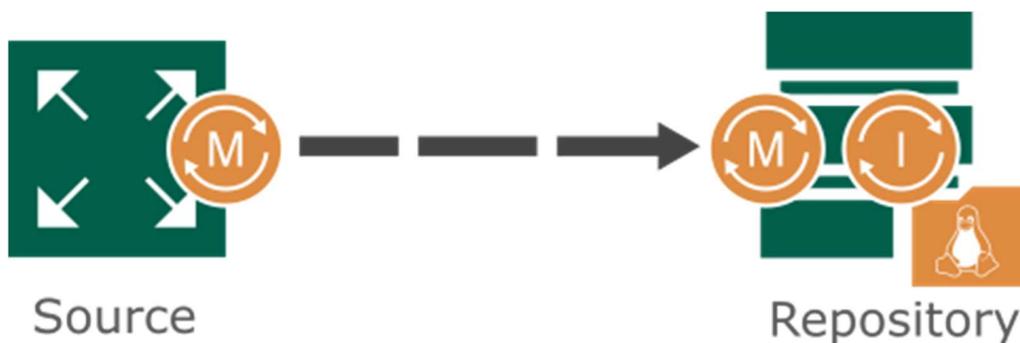
Here is a list of common acronyms found in this document as reference:

- RHEL: RedHat Enterprise Linux, the Linux version we will use in this document
- VHR : Veeam Hardened Repository. A Hardened Repositories distinguishes itself from a regular repository by being more secure and not using regular username and password to authenticates. This mitigates or lessens the likelihood of it being hacked
- VBR : Veeam Backup & Replication
- NIST (800-171) : A set of guidelines produced by the United States National Institute of Standards and Technology to assist in setting up safe and secure information systems.
- LUKS : The Linux Unified Key Setup (LUKS) is a disk encryption specification. LUKS is used to encrypt a block device (eg Hard Drives)
- NFS : Network File System (NFS) is a distributed file system protocol allowing you to access file on a remote system. This is typically used in Unix/Linux systems to share files over the system.
- NTP : Network Time Protocol. A protocol that servers and computers used to keep their local system clock up to date with other external servers.
- SSH : Secure Shell. Commonly used protocol to run commands on remote Unix/Linux systems over a secure tunnel.
- UUID : Universal Unique Identifier is a 128-bit value used to uniquely identify an object. The number is generated randomly but due to it sheer size (128 bit) the chance of generating the same number is extremely small.

## How Veeam Hardened Repositories work

Veeam Hardened Repositories (VHR) are Linux machines that can be installed by a customer preferably on a physical machines. They do not ship as an appliance but rather rely on existing Linux distributions. Organizations like RedHat or Canonical focus on maintaining Enterprise Linux distributions. Whenever there is a security issue or a new zero day exploit is found, they are the first to send out updates and fix the issue. For a hardened repository, it is critical that you can get the latest updates at any time.

For organizations already using for example RHEL, setting up a hardened repository does not require training and the systems can be deployed on existing infrastructure and processes already in place, rather than maintaining another system.



### *Processes VHR*

The VHR works by deploying the software over SSH. During this installation, two important components are installed: -

#### **The Veeam Data Mover (M):**

This process will be running under an unprivileged user who can only write backup files to a certain directory. The user should be stripped of any other rights after the installation is completed. During the backup process, it will be responsible for getting the data from a source workload (eg. Veeam VMware Proxy, Veeam Agent for Windows or Linux, etc.) and creating the backup files.

#### **The Immutability Service(I):**

This process will lock and unlock backup files while they are stored on the repository. This process is running under root to set and removes the locks but is not connected to the network. The locking mechanism is based on accurate time keeping on the server.

It is important to understand that locking these files means that we cannot modify them to apply backup retention policies. Strategies like Forever Incremental(.vib) and Reverse Incremental(.vib) do not work.

We must setup a full backup(.vbk) either synthetic or active to kickstart a new backup chain on a regular basis. Retention will then be applied when the backup files fall out of retention and are unlocked by the immutability service



### *XFS Reflink*

It is highly recommended to use the XFS filesystem when deploying a VHR. Strictly speaking it is not necessary but XFS can reduce the storage consumption and I/O requirements when creating a synthetic full. With XFS, when a synthetic full is created, the system does not read or write the blocks but instead uses Data Block Sharing (Reflink) capabilities of XFS. As the name implies, it links the synthetic full data blocks to the blocks that are in use by the existing backup files. This reduces storage and I/O requirement by only copying pointers (metadata) instead of complete blocks.

### Paranoia level

In this document we assume a high level of paranoia. For example a recommendation to shut down SSH might seem to be exaggerated for some customers or unworkable. However since this guide is geared towards Windows admins, it might be easier and secure to shutdown the service instead of hardening it for limited use.

If you deviate from these recommendations, make sure that you understand what you are doing and what are the risks.

### Consult all resources

Security best practices constantly evolve as attacks become more elaborate. In this document, we will limit the scope to Veeam/RedHat interaction. It might be a good idea to validate the RedHat Security Hardening guide and consult your security engineer.

For example setting up a BIOS/EFI security is something we do not discuss here but it can create another layer of security. Multi-factor authentication might also be an extra way to secure logins when you are enabling SSH to do remote maintenance etc.

- [RedHat Security Guide RHEL 8](#)

### Encryption

Encryption can be done on different levels. You can encrypt the data on a disk by using LUKS as described in the RHEL Security Hardening guide. However, Veeam Backup & Replication also supports encrypting the data at rest. Using Veeam Backup & Replication allows you to centrally manage the encryption keys.

You can refer to the Veeam documentation here:

- [Backup Job Encryption](#)

Encryption might also help in the event of a (physical) theft. Do keep in mind that encrypting the data does not protect you from a Ransomware attack as hackers might just re-encrypt your data with another key.

The goal of this document is to prevent that from happening in the first place.

## Command line style

We have tried to write all configurations in a command line style instead of instructing the user to edit configuration files. This should allow you to automate as much as possible which is extremely important for consistency.

In general, it is best to execute line by line if you are executing these commands directly by copy pasting from this document.

However the trade-off is that some commands might seem confusing. We have tried to balance automation and readability.

If you are new to the world of Linux bash scripting, you might want to review the following common patterns we will use through this document.

If you already have some Linux experience, you may skip this section.

```
MYVAR="Setting my var"  
echo $MYVAR
```

The first line sets up a variable called MYVAR. Afterwards we can refer to this variable by using the dollar sign, e.g. \$MYVAR. The use of variable is used because often certain text is repeated and by using a variable, the commands can be shortened. Also by defining the variable at the top, it easy to understand which part of the script you might want to adapt for your specific case.

```
echo "MY TEXT" | sed s/MY/YOUR/
```

The sed command is used frequently to replace a source text (in this example "MY") with a target string (in this case "YOUR"). Also interesting is the pipe sign |. It takes the output from the echo command and supplies it to sed as input. You can say that it chains the commands together. This chain of commands should output "YOUR TEXT".

```
cat <<EOF > my-file.txt  
line1  
line2  
EOF
```

This common pattern of using the cat command together with EOF allows you to write multiple lines to a single file. The result of this command should be a file called my-file.txt with two lines in it "line1" and "line2". When you copy paste these commands, you can copy

paste the whole block starting from cat to EOF as bash will wait for EOF to occurs before executing. That means there is no risk that certain parts are skipped because bash already started executing before copy pasting.

```
sudo systemctl start sshd
```

Sudo (super user do) is a special command that takes another command and executes it as root (main Linux administrator account). This would be equivalent to be using runas with administrator credentials on Windows.

In this example we use the systemctl to start the sshd service. We will systemctl multiple times to manage the services on the linux system much like services.msc on windows

```
alias secho="sudo echo"
```

Alias allows you to create a new command that will work as a prefix. In this example the alias secho is created. When a user execute secho "hello world", the system will actually execute substitute secho with "sudo echo". In this case this would result in sudo echo "hello world" would be executed.

If you have to retype the same command over and over again with the same parameters, this will shorten the amount of characters you need to type.

```
ls -alh /home
cd /home/veeamadm
echo "hello world" > newfile.txt
ls -alh
cat newfile.txt
rm newfile.txt
```

If you have never used Linux, it might be handy to know these commands to navigate the Linux file system. 'ls' is a command to list files in a certain directory. The parameter -a show all files (also hidden files starting with a dot), l specify to create a list and h prints a human readable size (instead of bytes). The home directory is the directory where typically users home directories are stored (compare to c: under windows). With cd we can go into a certain directory. The echo and > combination generates a new file with hello world as it's content. The cat command allows you to print the content of a file. Finally, the rm command allows you to remove files.

```
echo "my text" | sudo tee -a /etc/myfile
```

This command appends "my text" to the /etc/myfile. By using this syntax with sudo we can add a single line to a config file but executing it as an administrator thus editing files even if we don't have permissions to them.

```
nano newfile.txt
```

We will not use nano in this guide but if you are new to the command line, nano might be the easiest method to inspect and edit files. Nano using commands that are showing at the

bottom of the screen. Eg the interface will show “^x Exit” at the bottom. This means that if you use ctrl+x on the keyboard, nano will exit.

## Management Considerations

### Hardware vs VM

We strongly recommend that you use a physical server as a repository. Keep in mind that a hypervisor might introduce other vectors of attack. For example, you might have the most secure VM but if somebody would have access to the hypervisor, they could just delete the complete machine.

If you are new to Linux, you might wish to test it first in a VM, this may remove the initial barrier of setting up such systems and might provide an excellent Proof of Concept.

### Remote management

We strongly recommend physically disconnecting remote management for physical servers (eg iDRAC,IMM,iLO). For some companies this might be undesirable with data centers managed from a huge distance. However, this again adds a vector of attack.

If you do not disconnect the remote management, make sure to change the default password and to separate the network completely.

### Physical access

Make sure you limit the physical access to the server and that you keep track of those that could potentially have access to your immutable repository. A server might be a fortress in your network but if somebody has physical access, it can still be compromised.

It is important to note that malicious parties will primarily focus on destroying the backup data and encrypt the production environment. Removing just a couple of drives out of a complete RAID group might be just enough to make your backups inaccessible.

## Install

### Hardware setup

In general it is recommended to have a typical x86\_64 server which is designed to work as a “storage server”. These are servers that allow you to present a huge amount of internal disks in a very dense way.

CPU and RAM should be calculated using the methods described in the Veeam Best Practices [Repository Design on bp.veeam.com](https://www.veeam.com/best-practices-repository-design.html). There are no specific considerations for the VHR as the locking mechanism does not require any substantial resources.

Storage recommendations are in general similar to a regular repository:

- 2 SSDs or hard drives in RAID 1 for installing the OS. Since the installation is minimal, 128GB or more should be ample
- For the backup storage please use RAID controller that support RAID 60 as recommended by the best practices. Alternatively use RAID 6 if your RAID controller does not support RAID60. More recommendations can be found in the best practices [Block Storage \(DAS/SAN\)](#)

Avoid any external storage like iSCSI or FC. Although technically this could work, this would introduce other security vectors as the storage network could be compromised.

NFS storage can not be used to built a VHR. Not only would this not be recommended giving the previous statement but it would be impossible to setup a real VHR with immutable backups as immutable flags can not be set on backup files residing on NFS storage.

In terms of networking, it is strongly recommend to have at least a 10Gbit link or more for optimal performance. It is recommended to create a trunk for high availability. In Linux this is referred to as a team (older implementation bond).

During setup, a team or bond can be set up in the wizard. Consult your network engineer for a correct installation as the target switches need to be setup correctly to match the VHR network configuration. These settings can be changed after the install with the commands `nmtui` or `nmcli` but it is easier to use the GUI during the initial configuration. A static IP is strongly recommended.

Finally, you can disable IPv6. Veeam Backup & Replication v11 does not support IPv6 for data movement.

*Setting up the networking teaming/bonding is out of scope of this white paper because the configuration heavily depends on the target switch capabilities. Consult RedHat resources linked below:*

- [Configuring network bonding Red Hat Enterprise Linux 8 | Red Hat Customer Portal](#)
- [If You Like Bonding, You Will Love Teaming | Red Hat Blog](#)

## Installation type

For any installation, the ISO install is generally the easiest to obtain and install. Refer to your server manual for remote mounting or alternatively burn the ISO to a physical disk.

For current compatibility RHEL 8.5 iso (“rhel-8.5-x86\_64-dvd.iso”) was used. Make sure to validate checksums with those supplied by RedHat. When starting the installation, opt to do checksum before installing.

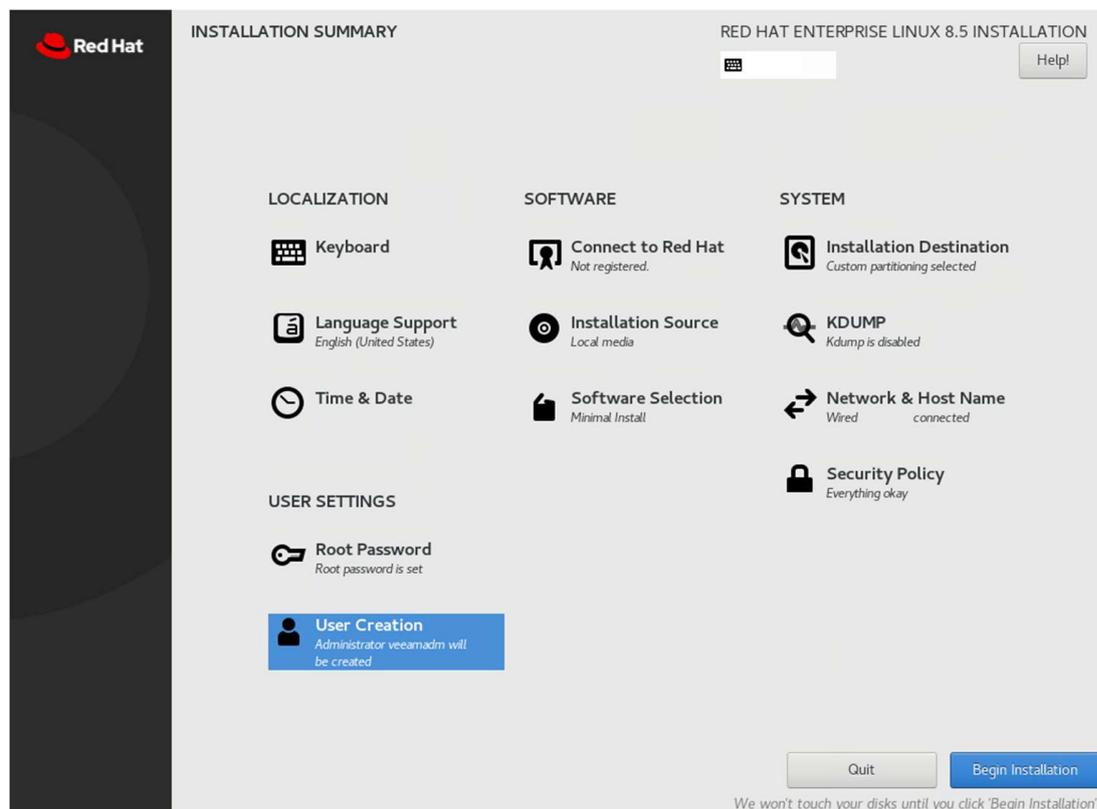
For the most secure repository type, please install a minimum installation as recommended by RedHat. There is only one software package required after installation

- [Customizing your installation Red Hat Enterprise Linux 8 | Red Hat Customer Portal](#)

## Begin Install

In the following sections we will describe the most common configuration points you must configure to begin the installation of your VHR. Certain points like Keyboard or Language Support are not discussed as they are user specific and do not impact security.

Once you have configured everything correctly, you should see something similar as the screenshot below and you are ready to install.



## Time Configuration

Make sure to configure your time zone correctly. Enabling (or disabling) NTP is a topic of discussion. If you have the ability to use a GPS dongle for timing, you eliminate another vector of attack. You can also choose to disable NTP but it is critical that you make sure the server is keeping correct track of time and that you verify the time on a regular basis.

You might want to configure NTS (Network Time Security) but at the time of writing the protocol is new and well known servers are limited. At this point it is also conflicting with the NIST Profile that we will select later. You can find a work around on the Chrony website but it will break the compliancy and therefore we will not document it. [NTS and NIST Profile](#)

## Software selection

To decrease the amount of attack vectors, the VHR should be installed with “Minimum Install” software selection. This will also allow you to select the NIST security profile.

## NIST 800-171 Security Profile

During install, you can select a security profile and validate if your installation is matching. For this document, we have selected the NIST 800-171 profile. The reason for selecting this profile is because it also automatically hardens your setup even further. These rules best practices were designed by the US government to make sure their IT systems are well protected. For example, it forces TMUX when you login. This is a screen multiplexer that will automatically lock itself in the case of inactivity. It forces FAPolicy which only allows binaries to run when they are known by the system. It sets up minimum SSL/TLS requirements for applications. All of this comes at the cost of complexity but it does setup a more secure system. For example, to apply this profile we need to setup the storage system in such a way that certain mount points are separated in logical or physical volumes.

Contrary to Windows, every drive or partition in Linux is mounted under the root level directory. The root directory “/” is typically the main disk. Compare this to the c: drive under most typical windows systems. Other drives or partitions are mounted under the root directory. For example, if you have a data drive, you might mount it under “/mnt/data” instead of a typical “d:” drive. Typically SWAP (extended memory) space is written to a separate disk. Compare this to Windows where this data is stored in the pagefile.sys

Here is a sample setup but you can deviate from these. In general, the Veeam software is extremely lightweight and does not require a lot of disk space.

To create such a setup, make sure to use manual partitioning. You can pre-create the LVM setup to remove some of the manual configuration

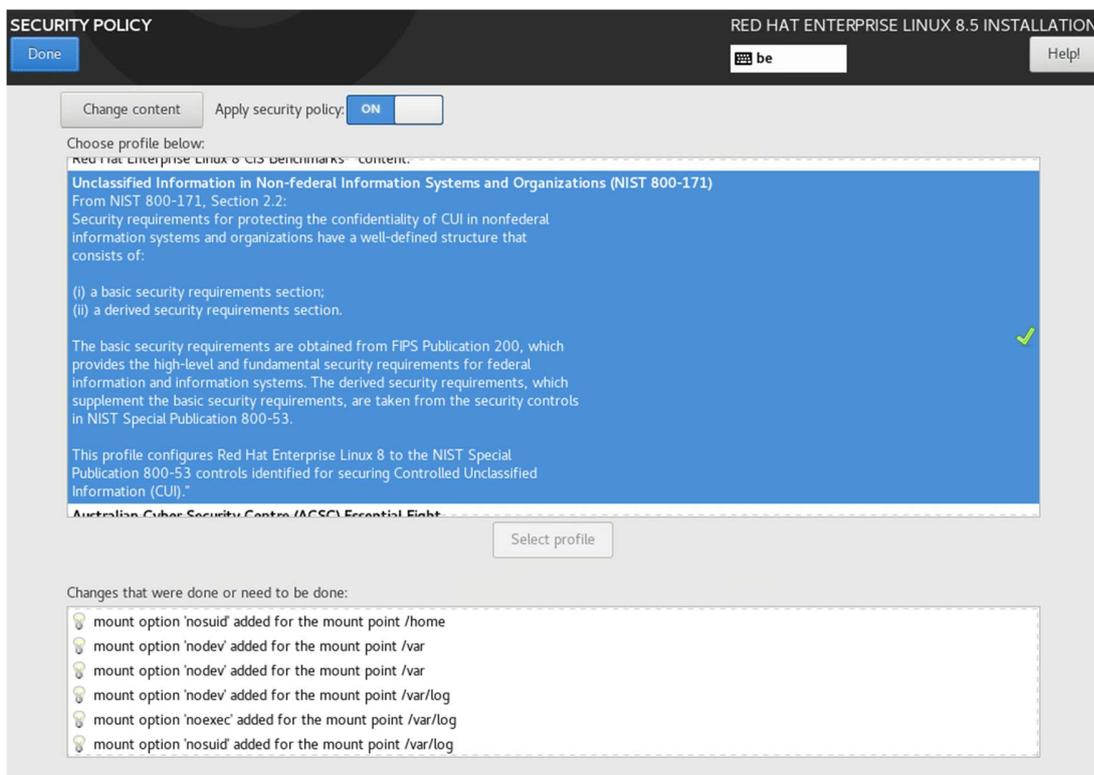
| Mount Point | Size                 |
|-------------|----------------------|
| /home       | 10 GiB               |
| /           | 10 GiB+ (remaining)  |
| /boot/efi   | 600 MiB (by default) |

|                |                      |
|----------------|----------------------|
| /boot          | 1 GiB (by default)   |
| Swap           | 4 GiB (by default) * |
| /var           | 5 GiB                |
| /var/log       | 5 GiB                |
| /var/log/audit | 10 GiB               |
| /var/tmp       | 5 GiB                |
| /tmp           | 10 GiB               |

\* Because the server will not hibernate, RedHat recommends 4GB as SWAP space

## [Security hardening Red Hat Enterprise Linux 8 | Red Hat Customer Portal](#)

During the setup select the NIST 800-171 profile, as shown in the screenshot below



## User setup

Make sure that you use a very strong (long) password and make sure you create a new separate non root user and designate it as administrator. The root user is the main admin account for Linux. As such, hackers always try to gain “root access” as this means they have unlimited access to the system. It is recommended to select a user name that is less common preferably something unique, e.g. avoid admin but use veeamadm.

**Avoid using the root user completely. Do not use it for login and configure an extremely long and complex password.**

## After Install

### Securing your admin user with key based authentication

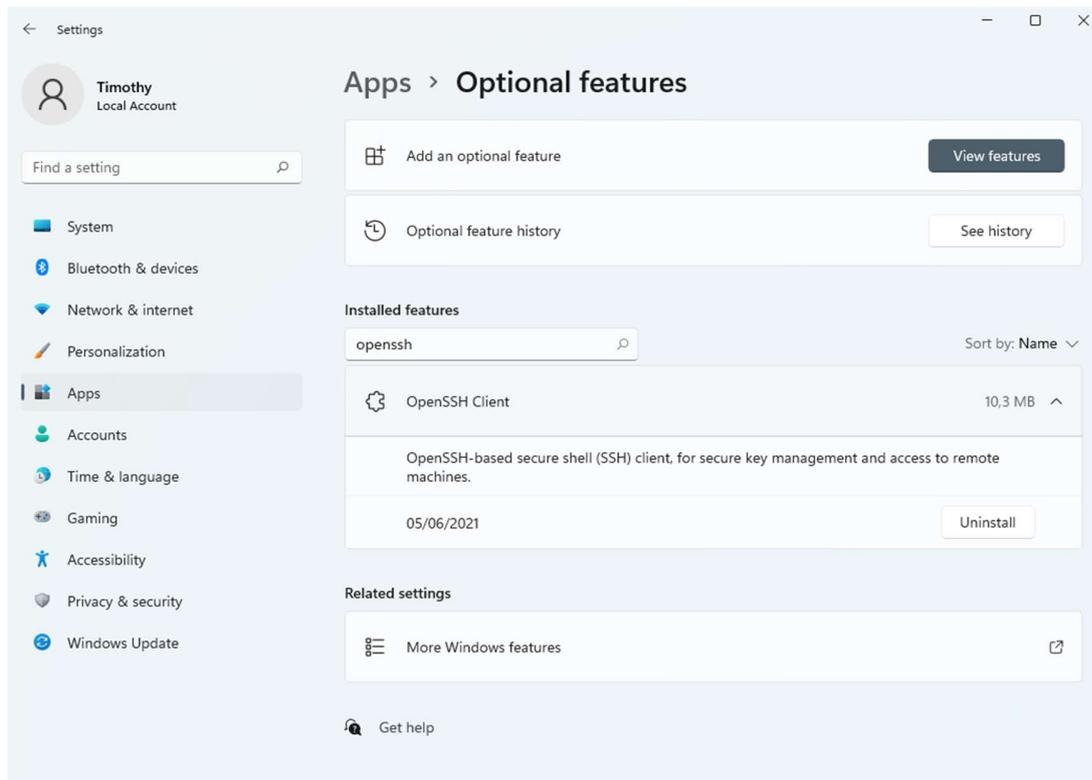
We recommend disabling SSH and even if you do so, it is still recommended to use key authentication, this makes sure that if you do enable SSH (temporarily), you are doing it in a secure way.

In general, you should have proper key management in place but this out-of-scope for this document. We will discuss how to generate a simple public/private key on your laptop if you have not such key management in place.

If you are new to key authentication, you can use `ssh-keygen` to generate a private/public key pair. The idea behind public/private keys is that you generate a key pair that is mathematically connected to each other. The private key is kept on the workstation as a secret and will encrypt the data. With the public key, the receiver (in this case the VHR) is then able to decrypt the data and ensure that the sender was the holder of the private key.

This kind of encryption is called asymmetric encryption and is different from symmetric encryption where both parties share the same key. It is important to understand that once the private key is compromised, a hacker can pretend to be the authentic source and thus it is important to password protect this key.

Since Windows 10, this tool is included in the operating system but it has to be installed as an optional package. You can find it under the name "OpenSSH Client" under the optional package feature. Make sure to install it before continuing.



### *OpenSSH Optional features*

Once you have installed OpenSSH, run this command on **the workstation** that you want to use to administer your repository remotely

```
ssh-keygen -t rsa -b 4096
```

**Do not store the key without a passphrase! It is recommended to check with your security officer which algorithm should be used. We used RSA/4096 key. You should get a similar output**

```
PS C:\Users\Timothy>
PS C:\Users\Timothy>
PS C:\Users\Timothy>
PS C:\Users\Timothy>
PS C:\Users\Timothy> ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\Timothy\.ssh\id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\Timothy\.ssh\id_rsa.
Your public key has been saved in C:\Users\Timothy\.ssh\id_rsa.pub.
The key fingerprint is:
SHA256:/1na6TdGxrtiIdTjWweVL7tLT4BKbjz6cJDb/u4tndo timothy@
The key's randomart image is:
+----[RSA 4096]-----+
|
| .
| ..
| .o
| . .o.+
| S . . +=o
| O . .B+
| o X .*o*
| * o.@+o
| ..oX+E++
+----[SHA256]-----+
PS C:\Users\Timothy>
```

### *Generating a key*

You can then copy the public key to your repository. **Run the following code on your workstation using powershell. You can find powershell via the start menu.** Make sure to replace the variables \$VEEAMADMINUSER and \$REPOIP to match your setup. Notice that powershell uses the dollar sign for both assigning and referring to a variable contrary to Bash scripting used on Linux.

```
$VEEAMADMINUSER="veeamadm"
$REPOIP="192.168.0.2"
$PUBKEYDATA=$(get-content $home\.ssh\id_rsa.pub)
ssh -l $VEEAMADMINUSER $REPOIP ("mkdir -p ~/.ssh && echo {0} >> ~/.ssh/authorized_keys" -f $PUBKEYDATA)
```

Once you have uploaded the key, you can connect remotely with ssh **from your workstation**

```
ssh -lveeamadm repository.fqdn.or.ip
```

```
https://aka.ms/powershell
Type 'help' to get help.
```

```
A new PowerShell stable release is available: v7.2.4
Upgrade now, or check out the release page at:
https://aka.ms/PowerShell-Release?tag=v7.2.4
```

```
PS C:\Users\Timothy> $VEEAMADMINUSER="veeamadm"
PS C:\Users\Timothy> $REPOIP="192.168.0.22"
PS C:\Users\Timothy> $PUBKEYDATA=$(get-content $home\.ssh\id_rsa.pub)
PS C:\Users\Timothy> ssh -l $VEEAMADMINUSER $REPOIP ("mkdir -p ~/.ssh && echo {0} >> ~/.ssh/authorized_keys" -f $PUBKEYDATA)
The authenticity of host '192.168.0.22 (192.168.0.22)' can't be established.
ECDSA key fingerprint is SHA256:FwqLUwQT1mRPOSErstXAwsakaop2Loa8F54rZQivkCI.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.22' (ECDSA) to the list of known hosts.
\S
Kernel \r on an \m

veeamadm@192.168.0.22's password:
PS C:\Users\Timothy> ssh -l $VEEAMADMINUSER $REPOIP
\S
Kernel \r on an \m

Enter passphrase for key 'C:\Users\Timothy\.ssh\id_rsa': |
```

### Copying the key

If you have generated the key pair on an alternative location, you can use the -i parameter

```
ssh -lveeamadm -i C:\Users\Timothy\.ssh\id_rsa.pub repository.fqdn.or.ip
```

### Initial update

In general it is a good idea to update the system to the latest patches. Keeping the system up to date is critical. This will require a valid RedHat subscription. If you have not done so during setup execute.

```
sudo subscription-manager register
```

Attach a license to your server

```
sudo subscription-manager attach --auto
```

The minimal install misses some software packages that is required to do the VHR installation. Please run the command below to install them. \* lsof : The lsof (list open files) package allows the installer to list open files and connections and validate if the installation was successful. \* attr : Allows us query the file attributes of the backup files. Strictly speaking, this package is not required but might be handy if you want to monitor the backup files afterwards

RHEL 8 uses the package manager “yum” to install/update/remove software. RedHat is in the processing of using a new package manager called “dnf” which takes similar input parameters.

```
sudo yum install -y lsof attr
```

Finally run an update to make sure the system is completely up to date.

```
sudo yum upgrade -y
```

Notice that the upgrade command might remove some unnecessary packages but at this stage it is less important.

## Automatic updates

You might consider automatic updates to keep your system up-to-date automatically. This comes down to a personal choice. An update might break your system but the question is do you balance stability over security. This setup might be useful if you are not able to access the system on a regular basis but you still want to have an up-to-date system.

You can also choose to only download but not install the updates with dnf-automatic. The example enables automatic download and install.

To use dnf-automatic, we first need to install it

```
sudo yum install -y dnf-automatic
```

Enable the automatic download and install option with

```
sudo systemctl enable dnf-automatic-install.timer
```

The installation is now 'enabled' to start at boot . To activate it directly, use the start command

```
sudo systemctl start dnf-automatic-install.timer
```

For more information, refer to the official documentation. For example, you might configure dnf-automatic to send out emails instead of logging to stdio.

- [Managing software packages Red Hat Enterprise Linux 8 | Red Hat Customer Portal](#)

## Enabling firewall

With a minimal install, RHEL does not enable the firewall to block most traffic. It is critical to enable the firewall to block unwanted access on all ports.

In this document we will use firewalld because this is one of the supported mechanisms by Veeam Backup & Replication 11 (firewalld, iptables or ufw). By default firewalld is "masked" so it is hidden from users. This is because the default firewall in RHEL is netfilter but this is not officially supported by Veeam. We have to unmask it first before we can enable it at boot and finally start it.

```
sudo systemctl unmask firewalld
sudo systemctl enable firewalld
sudo systemctl start firewalld
```

Firewalld uses a system of zones where groups of endpoints (for example IP ranges, data coming over specific interfaces) are given access to certain resources (ports). The default zone that is assigned to the network card is “public”.

```
sudo firewall-cmd --get-default-zone
sudo firewall-cmd --list-all
```

You can also remove the services we do not need. IPv6 is not supported by Veeam Backup & Replication v11. Cockpit is a remote management tool but you might want to avoid using it since it is another attack vector. By removing it from the default zone, access will be blocked for all other machines in the network.

```
sudo firewall-cmd --permanent --remove-service=cockpit
sudo firewall-cmd --permanent --remove-service=dhcpv6-client
```

We can also tweak SSH so it will only accept connections from the backup server or another management server or workstation.

**Watch out if you are configuring the system over SSH, you might lose your connection if you make a mistake.**

The following script will create a new SSH zone for the **backup server and the management stations**. You can add more IP’s by adding them between the brackets and separating them with a space. The zone is called “tmpssh” as we will remove at the end of the document to block remote access completely.

```
SSHSERVERLIST=(192.168.0.101 192.168.0.102)
```

```
sudo firewall-cmd --permanent --new-zone=tmpssh
```

```
alias tmpssh="sudo firewall-cmd --permanent --zone=tmpssh"
```

```
tmpssh --set-description="Allow ssh management from certain IPs"
tmpssh --set-short="Temporary Zone for SSH"
tmpssh --add-service=ssh
```

```
for s in ${SSHSERVERLIST[@]}; do
  tmpssh --add-source="$s/32"
done
```

Now we can remove SSH access from the main zone. This will ensure that SSH access will only be allowed to those systems that are in the “tmpssh” zone

```
sudo firewall-cmd --permanent --remove-service=ssh
```

If you are on SSH, you might want to review the folder “/etc/firewalld/zones/” before continuing. You can use ls to list the files and cat to show the content of the xml files. You can edit them with “sudo nano” if you feel there is a mistake.

```
sudo ls /etc/firewalld/zones
sudo cat /etc/firewalld/zones/public.xml
```

With “-permanent” we make changes to the config on disk but it will not impact the live system. We need to reload it from disk in memory (as would happen at boot). After we reload we can do a sanity check. Make sure only the public and the tmpssh zones are active.

```
sudo firewall-cmd --reload
sudo firewall-cmd --list-all-zones
```

## Adding a user that will act as a data mover

The Veeam process that will create backups has to run under an unprivileged user without sudo rights. This is why we will create a new user eg. veeamdatamover. Feel free to change this to a username but be consistent in the following chapters

```
VDMUSER=veeamdatamover
```

Create the user by using useradd. The -m parameter specifies to create the home directory under /home.

```
sudo useradd -m $VDMUSER
```

At this moment VHR only supports password authentication while deploying it from Veeam Backup & Replication. Start by generating a semi random password with. We can do this by reading 50 characters from the /dev/urandom device. This device is a virtual device that keeps emitting random characters. We will pipe the data to base64 so that only a limited amount of characters are used. Notice that the final 16 defines the amount of characters that will be used. You can choose a shorter password by changing 16 to for example 8.

```
RANDPASS=$(head -c 50 /dev/urandom | base64 | head -c 16)
```

You can check the password with the following command. By using less, the output will only be shown while the less command is active. When you add the repository to Veeam Backup & Replication, you will need this temporary password.

```
echo $RANDPASS | less
```

Use q if you are stuck in less

Now set the random password we generated

```
echo $RANDPASS | sudo passwd --stdin $VDMUSER
```

## Preparing your backup volumes

To identify your disk that you want to use as a backup target you can use the following command.

```
sudo lsblk -p -f
```

```
[veeamadm@vhr ~]$
[veeamadm@vhr ~]$
[veeamadm@vhr ~]$
[veeamadm@vhr ~]$
[veeamadm@vhr ~]$
[veeamadm@vhr ~]$ sudo lsblk -p -f
NAME                                FSTYPE LABEL UUID                                MOUNTPOINT
/dev/sda
├─/dev/sda1                          vfat      7767-F6D8                                /boot/efi
├─/dev/sda2                          xfs       3589ab04-dea1-4823-8429-db2db363c325    /boot
├─/dev/sda3                          LVM2_m    Yety88-KT5j-dKJU-IWmd-aGEr-EWwa-PYVeTQ /
│   ├─/dev/mapper/rhel-root          xfs       1e3d6b76-dee5-4dea-b403-616208c54a15    /
│   ├─/dev/mapper/rhel-swap         swap      fb351a0a-fc6f-46c1-aa69-7942a39a78c3    [SWAP]
│   ├─/dev/mapper/rhel-tmp          xfs       5a2e5821-477a-433a-b30a-c95973f63080    /tmp
│   └─/dev/mapper/rhel-var_log_audit
│       ├─/dev/mapper/rhel-var_log_audit xfs       c98abb50-88fc-4e66-9ab5-ee645bb045a0    /var/log/a
│       └─/dev/mapper/rhel-var_log
│           ├─/dev/mapper/rhel-var_log xfs       9da64d15-0db9-4cbb-b007-014ac9f91769    /var
│           └─/dev/mapper/rhel-var_log xfs       025ad365-8e24-4d78-97de-91b35b606b37    /var/log
└─/dev/mapper/rhel-var_tmp
    ├─/dev/mapper/rhel-var_tmp        xfs       83429f92-d07e-4f64-9073-243c6d48026e    /var/tmp
    └─/dev/mapper/rhel-home          xfs       89dd4d22-bc01-482f-9561-0596fb455766    /home
/dev/sdb
/dev/sr0
[veeamadm@vhr ~]$ |
[2] 0: sudo*                                     "vhr" 13:56 20-May-22
```

### Empty disk lsblk

In this example **/dev/sdb** is empty and ready for use. You can verify this by the fact there are no partitions under **/dev/sdb** and the FSTYPE column is empty.

Once you identified the disk, partition, format and mount it.

First define the variables correctly matching your setup. Remember that under Linux we don't use drive letters, but volumes are mounted under the root (/) filesystem. /mnt is a common place to mount disks that do not serve a Linux specific purpose.

```
VEEAMDISK=/dev/sdb
MOUNTPATH=/mnt/backup01
```

Create a primary partition on the disk. This partition will create a partition that will consume all the space. In general it is a best practice to create partitions instead of formatting the disk directly.

```
sudo parted $VEEAMDISK --script -- mklabel gpt
sudo parted $VEEAMDISK --script -- mkpart primary 0% 100%
```

Formatting with XFS and using the correct parameters is extremely important. These command is copied directly from the documentation so verify that they are still the same on [Fast Clone - User Guide for VMware vSphere](#)

```
sudo mkfs.xfs -b size=4096 -m reflink=1,crc=1 ${VEEAMDISK}1
```

Notice the **\${VEEAMDISK}1** parameter. This will make sure Bash understands that we want the content of VEEAMDISK variable concatenated to 1 referring to the first (and only) partition we created. In the document example this would yield **/dev/sdb1**.

To persistently mount the disk, we will need to add an entry to `/etc/fstab`. It is recommended to use the UUID of the disk instead of the disk name. You can extract the UUID in a variable with the following command. It uses the `blkid` command to print the UUID

```
DISKUUID=$(sudo blkid ${VEEAMDISK}1 -o value | head -n1)
```

Make sure the value in `$DISKUUID` is consistent with the output of `blkid`

```
sudo blkid ${VEEAMDISK}1
echo "Verify with : $DISKUUID"
```

```
[veeamadm@vhr ~]$ VEEAMDISK=/dev/sdb
[veeamadm@vhr ~]$ MOUNTPATH=/mnt/backup01
[veeamadm@vhr ~]$
[veeamadm@vhr ~]$ sudo parted $VEEAMDISK --script -- mklabel gpt
[veeamadm@vhr ~]$ sudo parted $VEEAMDISK --script -- mkpart primary 0% 100%
[veeamadm@vhr ~]$
[veeamadm@vhr ~]$ sudo mkfs.xfs -b size=4096 -m reflink=1,crc=1 ${VEEAMDISK}1
meta-data=/dev/sdb1          isize=512    agcount=4, agsize=3276672 blks
      =                       sectsz=512   attr=2, projid32bit=1
      =                       crc=1        finobt=1, sparse=1, rmapbt=0
      =                       reflink=1    bigtime=0 inobtcount=0
data      =                   bsize=4096  blocks=13106688, imaxpct=25
      =                       sunit=0       swidth=0 blks
naming    =version 2          bsize=4096  ascii-ci=0, ftype=1
log       =internal log      bsize=4096  blocks=6399, version=2
      =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime  =none              extsz=4096  blocks=0, rtextents=0
Discarding blocks...Done.
[veeamadm@vhr ~]$ DISKUUID=$(sudo blkid ${VEEAMDISK}1 -o value | head -n1)
[veeamadm@vhr ~]$ sudo blkid ${VEEAMDISK}1
/dev/sdb1: UUID="8218e856-b556-4e0f-8e01-763eb9212b4f" BLOCK_SIZE="512" TYPE="xfs" PARTLA
BEL="primary" PARTUUID="18f69b6d-3750-49af-81a7-2d9f210fef98"
[veeamadm@vhr ~]$ echo "Verify with : $DISKUUID"
Verify with : 8218e856-b556-4e0f-8e01-763eb9212b4f
[veeamadm@vhr ~]$ |
[2] 0-bash* "vhr" 14:20 20-May-22
```

### Verify UUID

Make a directory where we will mount the backup disk e.g.

```
sudo mkdir -p $MOUNTPATH
```

The `/etc/fstab` file is read at boot and defines which disks need to be mounted. In order to have our disk mounted at startup automatically, we need to add a definition to this file. Once we have added the line, we can use `mount` command to mount the disk manually

```
echo "UUID=$DISKUUID $MOUNTPATH xfs defaults,nodev,noexec,nosuid 0 0" | sudo
tee -a /etc/fstab
sudo mount -U $DISKUUID
```

It is a good idea to verify if the disk was mounted. The `grep` command looks for a certain text in the output, in the example the disk. If the output of this command is empty, something went wrong.

```
sudo mount | grep $MOUNTPATH
```

The datamover user we created in the previous chapter has to be able to write to the volumes you have created.

```
VDMUSER=veeamdatamover
```

Assign the permissions to the directory. The `chown` command changes the owner of the directory to our user and the group with the same name (which is automatically created when you create the user). The style `user:group` is used to define the new owners.

`Chmod` changes the access permissions for files. We will change the permission on the directory to `700`. The first digit defines the permissions of the user. The second digit the group permissions. The last digit represents the rest of the world.

Permissions are configured by adding up the numbers that correspond to the rights: - 4 = Read permission - 2 = Write permission - 1 = Execute permission

By adding  $4+2+1 = 7$  we assign all the permissions to the "user". A zero means no permissions are assigned. By using `700`, we define that the user (owner) can do anything on the directory but the rest of the world including the group that owns the directory have no permissions on this directory. The root user is a special case as it always can access these resources.

```
sudo chown -R $VDMUSER:$VDMUSER $MOUNTPATH
sudo chmod -R 700 $MOUNTPATH
```

Repeat these steps for every disk you want to mount but make sure to define another `MOUNTPATH` for every disk.

## Reboot

At this point it is a good idea to reboot the server. Strictly speaking, this should not be necessary but, it will validate.

- Updates did not break the server
- Volumes are correctly mounted (**`mount | grep /mnt/backup`**)
- Firewall is correctly configured (**`sudo firewall-cmd --list-all-zones`**)

You can reboot a RHEL server by using

```
sudo shutdown -r now
```

## Registering the repository

### Adding the repository

We can now add the repository with the username you have select and the random password you have designated. Set the variable if you have not done so

```
VDMUSER=veeamdatamover
```

Now since we took the NIST Profile, we have to follow the instruction in [KB4250: How to configure hardened repository on Red Hat Enterprise Linux 8 with NIST 800-171 security profile](#)

#### *Disable TMUX temporarily*

TMUX is a terminal multiplexer which allows you to create multiple terminals in one session (comparable how you could have multiple tabs in a webbrowser). It also locks the terminal automatically after a certain amount of inactivity. Unfortunately the Veeam installer does not see the TMUX output and therefore we must disable it temporarily as instructed by the KB. The following command first make a backup of /etc/bashrc and then live edits it. It does this by commenting out the lines that says tmux needs to automatically run once the user has logged in.

```
sudo cp /etc/bashrc /etc/bashrc.backup
sudo sed -i 's/\(.*exec tmux\)\/#TMUXDISABLE#\1/g' /etc/bashrc
```

#### *Sudo and umask settings*

We also need to apply some temporary sudo permissions and set umask to 0022. Basically, we temporarily give the VDMUSER sudo rights (admin rights) so the Veeam installer has enough rights to install the software. We must remove these rights after the installer is complete.

The Umask permission is specified by the KB article. It defines what are the default permission that should be assigned to newly created files. This is a complicated topic as they are the opposite of the regular permissions (e.g. 022 is the opposite of 755 thus 755 will be applied). For this document we will just follow the KB article steps.

```
printf "%s ALL=(ALL:ALL) ALL\nDefaults:%s umask_override\nDefaults:%s umask=0022\n" \  
$VDMUSER $VDMUSER $VDMUSER | sudo tee ~/90-veeamtmpadd  
sudo visudo -cf ~/90-veeamtmpadd
```

Make sure there are no “%s” patterns left in the output and that visudo validated the file. Here is an **example output**



```
if [ \${(\$(date +%s)%5)} -eq 0 ]; then
  printf "%s Waiting for server cert\n" "\$(date)";
fi
sleep 1s;
done
if [ -f "/opt/veeam/transport/veeamimmureposvc" ]; then
sudo fapolicyd-cli --file add /opt/veeam/transport/veeamagent
sudo fapolicyd-cli --file add /opt/veeam/transport/veeamtransport
sudo fapolicyd-cli --file add /opt/veeam/transport/veeamimmureposvc
sudo systemctl restart fapolicyd
else
echo Binaries not uploaded yet
fi
EOF
```

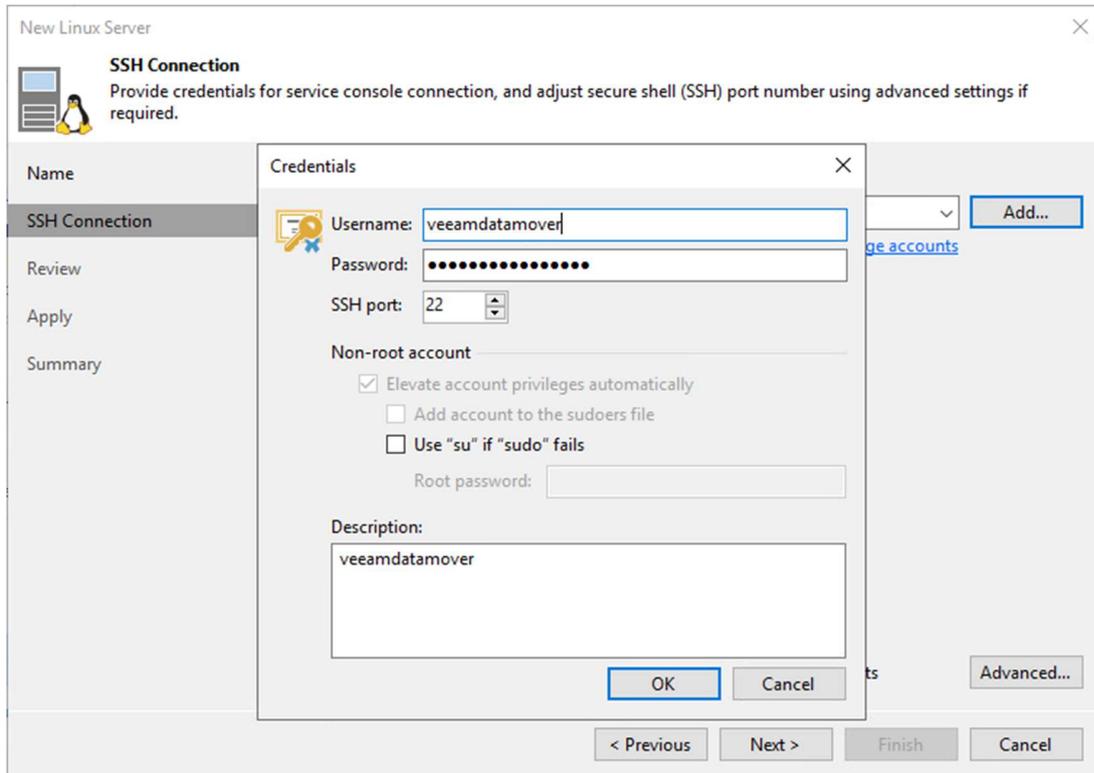
You can make the script executable by adding +x permissions

```
chmod +x tryfa.sh
```

When you are ready to add the repository to the Veeam Backup & Replication server, run the script with sudo rights. Keep it running, it should automatically stop once the server certificate step has ran and the binaries where added to the fapolicyd

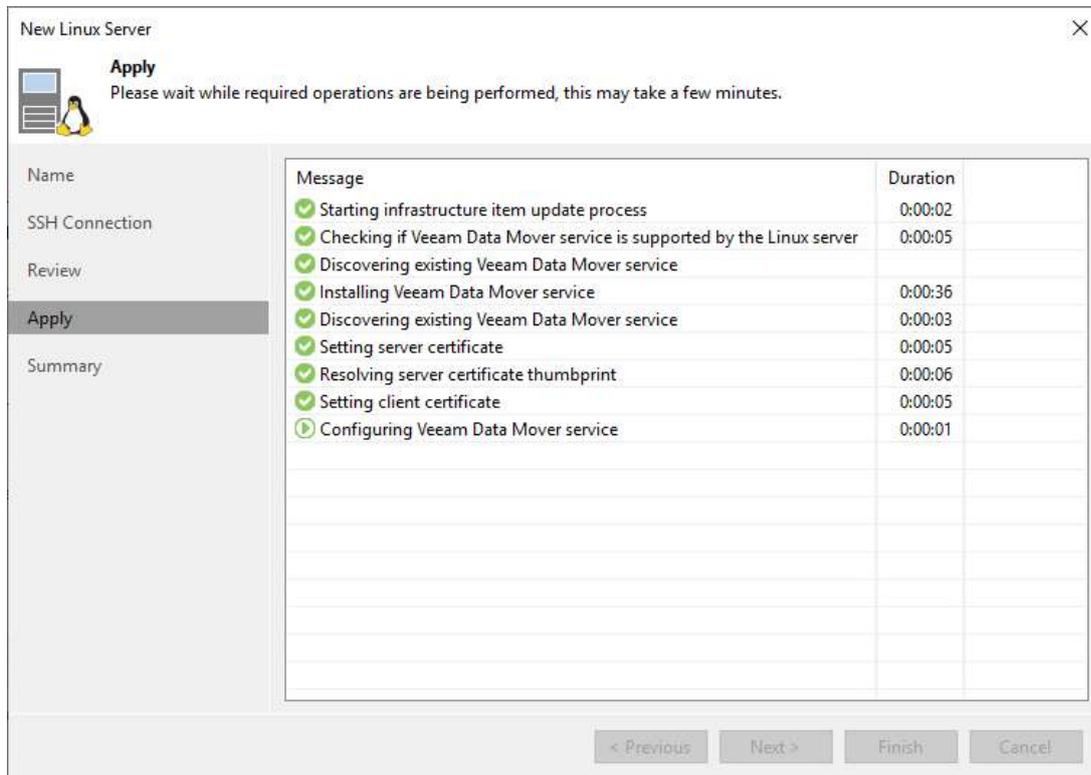
```
sudo ./tryfa.sh
```

You can now add the repository to Veeam Backup & Replication. Make sure to use **single use credentials** when adding a Linux repository.



*Use the data mover*

Veeam Backup & Replication will now try to upload the binaries and certificates it uses for authentication.



### *Certificate step passed*

During the certification upload step, the script should stop after it added the binaries to FAPolicyd

```

Fri May 20 14:55:45 CEST 2022 Waiting for server cert
Fri May 20 14:55:50 CEST 2022 Waiting for server cert
Fri May 20 14:55:55 CEST 2022 Waiting for server cert
Fri May 20 14:56:00 CEST 2022 Waiting for server cert
Fri May 20 14:56:05 CEST 2022 Waiting for server cert
Fri May 20 14:56:10 CEST 2022 Waiting for server cert
Fri May 20 14:56:15 CEST 2022 Waiting for server cert
Fri May 20 14:56:20 CEST 2022 Waiting for server cert
Fri May 20 14:56:25 CEST 2022 Waiting for server cert
Fri May 20 14:56:30 CEST 2022 Waiting for server cert
Fri May 20 14:56:35 CEST 2022 Waiting for server cert
Fri May 20 14:56:40 CEST 2022 Waiting for server cert
Fri May 20 14:56:45 CEST 2022 Waiting for server cert
Fri May 20 14:56:50 CEST 2022 Waiting for server cert
Fri May 20 14:56:55 CEST 2022 Waiting for server cert
Fri May 20 14:57:00 CEST 2022 Waiting for server cert
Fri May 20 14:57:10 CEST 2022 Waiting for server cert
Fri May 20 14:57:15 CEST 2022 Waiting for server cert
Fri May 20 14:57:20 CEST 2022 Waiting for server cert
Fri May 20 14:57:25 CEST 2022 Waiting for server cert
Fri May 20 14:57:30 CEST 2022 Waiting for server cert
Fri May 20 14:57:35 CEST 2022 Waiting for server cert
Fri May 20 14:57:40 CEST 2022 Waiting for server cert
Fri May 20 14:57:50 CEST 2022 Waiting for server cert
[veeamadm@vhr ~]$
[0] 0: bash* "vhr" 14:58 20-May-22

```

### *Script stopped*

Please refer to the user guide when in doubt

## Deploying Hardened Repository - User Guide for VMware vSphere

### Multiple physical servers

If you are adding multiple physical servers, this FAPolicy script might be cumbersome. Once you have done the first machine, you can copy paste the hashes from that system

```
sudo cat /etc/fapolicyd/fapolicyd.trust | grep /opt/veeam
```

For example for VeeamTransport\_11.0.1.1261 these are the corresponding hashes. Every update will have a different hash

```
/opt/veeam/transport/veeamagent 59768960  
27d0d1882a4410edc1dc859966237532f39ba6764f374b25c894cf064a6fd995  
/opt/veeam/transport/veeamtransport 8060352  
dbe923a8c59f5c8d77e0f3ff49e7a67a754cf65f1af10504b1f29c8576234098  
/opt/veeam/transport/veeamimmureposvc 5664064  
30eb8a894532f79991a41d555ad8c05abb22ff23c216b5be5a3f9410dcc00
```

You can then prepopulate fapolicyd.trust on other server and restart it

```
cat | sudo tee -a /etc/fapolicyd/fapolicyd.trust <<EOF  
/opt/veeam/transport/veeamagent 59768960 27d0d1882a4410edc1dc859966237532f39b  
a6764f374b25c894cf064a6fd995  
/opt/veeam/transport/veeamtransport 8060352 dbe923a8c59f5c8d77e0f3ff49e7a67a7  
54cf65f1af10504b1f29c8576234098  
/opt/veeam/transport/veeamimmureposvc 5664064 30eb8a894532f79991a41d555ad8c05  
abb22ff23c216b5be5a3f9410dcc00  
EOF  
sudo systemctl restart fapolicyd
```

## Post registration hardening

### Locking down after registration

In the previous section we defined a user variable that represents our datamover user. If you have lost the session, start by recreating the variable

```
VDMUSER=veeamdatamover
```

These steps are critical to have a secure repository and are not optional. First let's remove sudo permissions from the veeam data mover user. This is critical as this makes sure that even if the data mover process is compromised, it has very limited permission

```
sudo rm -f /etc/sudoers.d/90-veeamtmpadd
```

Next up, let's restore TMUX so that it will run whenever a new connection is established.

```
sudo sed -i 's/#TMUXDISABLE#//g' /etc/bashrc
```

Let's set a long random password so logins are no longer possible for this user

```
head -c 50 /dev/urandom | base64 | head -c 50 | sudo passwd --stdin $VDMUSER
```

Finally lock the account so interactive logons are not possible.

```
sudo usermod -L $VDMUSER
```

### Hardening SSH at startup

For higher security disable SSH (remote access) at startup after you have added the repository with. If you are running these commands over SSH, this might break the connection. In that case, execute the stop command after you have finished everything

```
sudo systemctl disable sshd  
sudo systemctl stop sshd
```

Check the status of the sshd service. Make sure it is no longer running

```
sudo systemctl status sshd
```

By default root login should be disabled over SSH but you can always make sure this is the case. We can check service it's config file (/etc/ssh/sshd\_config). It should have a line in there that says PermitRootLogin no.

```
sudo cat /etc/ssh/sshd_config | grep PermitRootLogin
```

Make sure there is no # sign in front of this line as this would comment out the statement

By default you can login with a username and password. Let's disable this with the following command so that only key (private/public) authentication is allowed.

In this scenario, make sure that you have generated keys and uploaded them to your admin (eg veeamadm) account if you want to execute remote management. To make such edit, we will first make a backup of the config file. Then we can edit the file with sed. Notice that during update, you might need to enable PasswordAuthentication again.

```
sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.backup
sudo sed -i 's/PasswordAuthentication.*$/PasswordAuthentication no/g' /etc/ssh/sshd_config
```

In this scenario, sed is looking for every file that has PasswordAuthentication and replaces everything behind it with no. This effectively disables PasswordAuthentication

Tip: During update, you need to re-enable PasswordAuthentication. You can do this by changing no to yes in the sed command line.

It is good idea to checkup that change has succeeded with

```
sudo cat /etc/ssh/sshd_config | grep PasswordAuthentication
```

Additional you can add multifactor authentication but this is out of scope of this document

## Secure the firewall openings after setup

You do not need SSH after the initial setup is done. You can therefore block access from the backup server. Remember that initially we made a temporary zone for SSH (tmpssh). By removing the backup server from that zone, we can remove SSH from the backup server.

```
BACKUPSERVERIP=192.168.0.101
sudo firewall-cmd --permanent --zone=tmpssh --remove-source=$BACKUPSERVERIP/32
```

Alternatively you can now remove the zone "tmpssh". This is more secure as it would make sure that even if somebody started SSH, it would not be accessible from the outside world.

```
sudo firewall-cmd --permanent --delete-zone=tmpssh
```

Reload the firewall to make the changes active

```
sudo firewall-cmd --reload
sudo firewall-cmd --list-all-zones
```

Tip: During update, you need to enable SSH. You can follow the initial setup steps to recreate the SSH firewall rules if you have removed the tmpssh zone

## Virtual offline repository

Firewall-cmd has an interesting mode which is called panic. It allows you to disconnect from the network in case of an attack. You could schedule this panic mode in the morning after the backups have finished (e.g. 8h00) and exit it just before the backups will start (eg 21h55). This creates another level of security that would make the repository unreachable during

office hours for example. It is important that you implement this configuration carefully. **Failure in correctly implementing this might lead to unexpected downtime.** Also notice that this will shutdown all connections, this means that remote management will not be possible during panic mode.

Finally, restoring from this setup would be more difficult because the repository is offline during the day. You should only implement this if you are able to **physically access** the repository quickly. You should also document a clear procedure on what needs to be done when you want to interact with the repository outside these windows

```
cat | sudo tee -a /etc/crontab <<EOF
55 21 * * * root firewall-cmd --panic-off
00 8 * * * root firewall-cmd --panic-on
EOF
```

Reload crontab to make sure the file is read

```
sudo systemctl reload crond
```

To disable panic mode manually during the day, run

```
sudo firewall-cmd --panic-off
```

## Hardening Chrony

If you did enable Chrony, it might be a good idea to set minsources for example to 3. This way, at least 3 servers need to agree before Chrony will adjust the time. Maxchange will make chrony ignore changes bigger than 1000 seconds. If this happens twice, chrony will stop to prevent further attacks.

```
sudo cp /etc/chrony.conf /etc/chrony.conf.backup.min
sudo sed -i -r 's/#?minsources.*/minsources 3/' /etc/chrony.conf
echo maxchange 1000 1 2 | sudo tee -a /etc/chrony.conf
```

Alternatively, you might consider using NTS (secure NTP servers). Unfortunately, the NTS RFC 8915 is fairly recent (end 2020). This means that the amount of available servers are limited. At the time of writing, NTS is conflicting with the NIST Profile therefore this document does not describe on how to configure it.

- [Chrony RHEL Manual](#)
- [Chrony – Frequently Asked Questions](#)
- [Chrony - NIST Compliance Error](#)

## Monitoring your Veeam Hardened Repository

### Monitoring disk usage

You can use disk usage with the `df` command. The `-h` parameter define human readable sizes instead of bytes. By using the `grep` command we can filter out our Veeam repositories

```
df -h | grep /mnt/backup
```

In this example you can see 7% usage (3.2G out of 50G)

```
[veeamadm@vhr ~]$ df -h | grep /mnt/backup
/dev/sdb1          50G  3.2G   47G   7% /mnt/backup01
[veeamadm@vhr ~]$ |
```

### Monitoring the Veeam Processes

To see the running process on linux you can use the `ps` command. By using `-e`, we can see every process. `-f` show the “full format” for every process (for example the file path is shown)

```
ps -ef | grep -e "/veeamtransport" -e "/veeamimmu"
```

You will be able to see three processes running. One is the `dataservice` that is running the transport service. It is running as the `veeamdatamover` (`veeamda+` shortened). The other two processes are the `run-environmentsvc` which dynamically opens the firewall during backup and the `immutability service` which lock the backup files.

```
[veeamadm@vhr ~]$ ps -ef | grep -e "/veeamtransport" -e "/veeamimmu"
veeamda+   1514      1  0 15:14 ?        00:00:03 /opt/veeam/transport/veeamtransport --
run-service
root      1773    1514  0 15:14 ?        00:00:00 /opt/veeam/transport/veeamtransport --
run-environmentsvc 7:6
root      1775    1514  0 15:14 ?        00:00:01 /opt/veeam/transport/veeamimmureposvc
--subprocess --log /var/log/VeeamBackup --stdio 9:7
```

```
ps -ef
```

The process also reveals other interesting data. We can see that the Veeam software is installed under `/opt/veeam/`. Also, the logging is done to a directory called `/var/log/VeeamBackup/`.

Finally you can also check the services via `systemctl`

```
sudo systemctl status veeamtransport
```

```
[veeamadm@vhr VeeamBackup]$ sudo systemctl status veeamtransport
● veeamtransport.service - VeeamTransport
   Loaded: loaded (/etc/systemd/system/veeamtransport.service; enabled; vendor preset: di
   Active: active (running) since Mon 2022-06-20 15:14:41 CEST; 7h ago
   Main PID: 1514 (veeamtransport)
     Tasks: 10
    Memory: 133.9M
    CGroup: /system.slice/veeamtransport.service
            └─1514 /opt/veeam/transport/veeamtransport --run-service
              └─1773 /opt/veeam/transport/veeamtransport --run-environmentsvc 7:6
                └─1775 /opt/veeam/transport/veeamimmuresvc --subprocess --log /var/log/Veeam
Jun 20 15:14:41 vhr systemd[1]: Starting VeeamTransport...
Jun 20 15:14:41 vhr systemd[1]: Started VeeamTransport.
```

*systemctl services*

## Checking the logs

If ever you need to troubleshoot the logs, they can be found under `/var/log/VeeamBackup/`. You can go in this directory by using the `cd` command. With `ls`, you can then list the log files

```
cd /var/log/VeeamBackup/
ls
```

Here is a sample log. You can see the generic services log and a specific subfolder for a Job that has run

```
[veeamadm@vhr ~]$ cd /var/log/VeeamBackup/
[veeamadm@vhr VeeamBackup]$
[veeamadm@vhr VeeamBackup]$ ls
Agent.LinuxFileCommander.1.log  VeeamAgent.GetContentInfoOperation.log
Agent.LinuxFileCommander.log    VeeamEnvironmentSvc.log
Backup_Job_2                    VeeamImmutableRepoSvc.log
Util.InfraItemSaver.log         VeeamTransportCli.log
Util.VolumesHostDiscover.log   VeeamTransportSvc.log
[veeamadm@vhr VeeamBackup]$
```

*ls /var/log/VeeamBackup*

You can also read a log with the `less` command or alternatively, you can use `nano` to go back and forward more easily. For example to read the Immutable log command, you can use the following command

```
less VeeamImmutableRepoSvc.log
```

You can navigate in `less` by using the space command to go down. Use the letter `q` to quit the application.

If you only want to see the last line of a log you can use `tail`. `Tail` has an interesting `-f` which will “following the log”. This might be handy during troubleshooting. It will keep showing the last 10 lines.

```
tail -f VeeamImmutableRepoSvc.log
```

## Analysing the backup files

We can analyse the backup files. However since we do not have the correct permissions, we have to escalate to root by using sudo. By supplying the -i parameter, we will start an interactive session. The prompt will change to [root@hostname].

```
sudo -i
```

**It is important to understand that you should use sudo -i very carefully. In fact you should avoid to use it at all time. We are showing it here for educational purpose. If you want to verify the system on a daily basis, you might use the immutability report below**

We can now enter our backup folder.

```
cd /mnt/backup01
```

You can use the find command to list all files. This will list all the vbk, vib etc. with all of it's locations

```
find
```

To show the data usage per job, you can use the du command. It will summarize the data per directory

```
du -h
```

```
[root@vhr backup01]# find
.
./Backup Job 2
./Backup Job 2/Backup Job 2D2022-05-23T133152_D3E3.vbk
./Backup Job 2/Backup Job 2.vbm
[root@vhr backup01]# du -h
2.8G  ./Backup Job 2
2.8G  .
[root@vhr backup01]#
```

### *Du and Find*

To see the regular permissions on the file, we can enter a job directory and list all the files

```
cd Backup\ Job\ 2/
ls -alh
```

```
[root@vhr backup01]# cd Backup\ Job\ 2/
[root@vhr Backup Job 2]# ls -alh
total 2.8G
drwxr-xr-x. 2 veeamdatamover veeamdatamover 145 Jun 20 22:10 .
drwx----- 3 veeamdatamover veeamdatamover 26 May 23 14:14 ..
-rw-r--r--. 1 veeamdatamover veeamdatamover 2.8G May 23 14:16 'Backup Job 2D2022-05-23T133152_D3E3.vbk'
-rw-r--r--. 1 veeamdatamover veeamdatamover 2.2M Jun 20 22:10 'Backup Job 2D2022-06-20T212505_E43B.vib'
-rw-r--r--. 1 veeamdatamover veeamdatamover 15K Jun 20 22:10 'Backup Job 2.vbm'
-rw-r--r--. 1 root root 243 Jun 20 22:10 .veeam.1.lock
```

### *List files*

These also shows a lock file which is unique for hardened repositories. This file is a just an XML file. You can analyse it with xmllint. For example using `-format` will prettify the XML output so it is more readable. Notice that the file might have a different digit in its name

```
cat .veeam.1.lock | xmllint --format -
```

```
[root@vhr Backup Job 2]# cat .veeam.1.lock | xmllint --format -
<?xml version="1.0"?>
<LockData>
  <File Name="Backup Job 2D2022-05-23T133152_D3E3.vbk" ImmutableTillUtc="2022-06-27 20:10:52" Finalized="true"/>
  <File Name="Backup Job 2D2022-06-20T212505_E43B.vib" ImmutableTillUtc="2022-06-27 20:10:52" Finalized="true"/>
</LockData>
```

### *xmllint on lock file*

Veeam stores the immutability date in 2 ways. One is the lock files, the other is by using extended file attributes. You can check those with `getfattr` on the VBK or VIB files.

```
getfattr -n user.immutable.until *.vbk
```

```
[root@vhr Backup Job 2]# getfattr -n user.immutable.until *.vbk
# file: Backup Job 2D2022-05-23T133152_D3E3.vbk
user.immutable.until="2022-06-27 20:10:52"
```

### *getfattr on vbk files*

To see how Veeam locks files, we can use the `lsattr` command to list the extended attributes

```
lsattr
```

```
[root@vhr Backup Job 2]# lsattr
----i----- ./Backup Job 2D2022-05-23T133152_D3E3.vbk
----i----- ./Backup Job 2D2022-06-20T212505_E43B.vib
----- ./Backup Job 2.vbm
[root@vhr Backup Job 2]#
```

### *Listing extended attributes*

You can see the “i” flag here on the vbk files which means they are immutable. The immutable flag is interesting because although the `veeamdatamover` user is owner of the files, it can not delete the files. Even the root user is not able to delete them unless it unlocks (removes the immutable flag first).

```
[root@vhr Backup Job 2]# lsattr
----i----- ./Backup Job 2D2022-05-23T133152_D3E3.vbk
----i----- ./Backup Job 2D2022-06-20T212505_E43B.vib
----- ./Backup Job 2.vbm
[root@vhr Backup Job 2]# rm Backup\ Job\ 2D2022-06-20T212505_E43B.vib
rm: remove regular file 'Backup Job 2D2022-06-20T212505_E43B.vib'? y
rm: cannot remove 'Backup Job 2D2022-06-20T212505_E43B.vib': Operation not permitted
[root@vhr Backup Job 2]#
```

### *Try removing the files*

If in the extreme event that you need to delete backup files, you can unlock this files as root with `chattr`. This shows how important it is to also make sure nobody can elevate to root. **By**

**unlocking the files, you are making them vulnerable for attacks. Only do this when you have no other choice**

```
chattr -i 'Backup Job 2D2022-05-23T133152_D3E3.vbk'
lsattr
```

```
[root@vhr Backup Job 2]# chattr -i 'Backup Job 2D2022-05-23T133152_D3E3.vbk'
[root@vhr Backup Job 2]# lsattr
----- . /Backup Job 2D2022-05-23T133152_D3E3.vbk
---i----- . /Backup Job 2D2022-06-20T212505_E43B.vib
----- . /Backup Job 2.vbm
----- . /-n
```

*Removing the immutable flag*

## Immutability Report

Remembering all these commands might be cumbersome. You can make a small overview for when you logon physically to terminal. Below you can find some sample code that monitors all of the directories under /mnt that start with backup, eg. /mnt/backup01 etc.

```
cat <<EOF > qc.sh
#!/bin/sh
BASEPATH=/mnt
SEARCH=backup

cdate=\$(date)

printf "Current Date: %s" "\$cdate"
printf "\n\nFS usage: \n"
df -h | grep "\$BASEPATH/\$SEARCH" | column -t -o " | "

printf "\n\nFiles:\n"
printf "%-20s | %10s | %s \n" "Immutability" "Size" "Backup File"
find \$BASEPATH -regextype posix-extended -regex "\$BASEPATH/\$SEARCH.*[.](vbk|vib)" -print0 | while read -d \$'\0' backupfile
do
    immudate=\$(getfattr -n user.immutable.until "\$backupfile" 2> /dev/null |
grep until | cut -f2 -d= | sed s/\\/\\/g )
    size=\$(du -h "\$backupfile" | cut -f1)
    printf "%-20s | %10s | %s \n" "\$immudate" "\$size" "\$backupfile"
done
EOF
chmod +x qc.sh
```

When ever you login to the terminal, you can now get a quick overview.

```
sudo ./qc.sh
```

You might want to pipe it to more in case you have a lot of backup files

```
sudo ./qc.sh | more
```

```
[veeamadm@vhr ~]$ sudo ./qc.sh | more
Current Date: Wed Jun 22 18:11:40 CEST 2022

FS usage:
/dev/sdb1 | 50G | 3.2G | 47G | 7% | /mnt/backup01

Files:
Immutability | Size | Backup File
2022-06-27 20:10:52 | 2.8G | /mnt/backup01/Backup Job 2/Backup Job 2D2022-05-23T133152_D3E3.vbk
2022-06-27 20:10:52 | 2.2M | /mnt/backup01/Backup Job 2/Backup Job 2D2022-06-20T212505_E43B.vib
2022-06-27 21:01:37 | 2.2M | /mnt/backup01/Backup Job 2/Backup Job 2D2022-06-20T221609_26F0.vib
2022-06-27 21:07:22 | 2.2M | /mnt/backup01/Backup Job 2/Backup Job 2D2022-06-20T222158_6137.vib
```

## Report

## Conclusion

Veeam Hardened Repository at first glance seems to be hard to set up but with this document we have shown that it is a manageable task for any administrator.

The advantages for such a repository are clear:

- Hardened repository that helps you in defense from cyber attacks
- No need for Windows licenses and their associated costs.
- A mature platform that is inherently more secure and performant. Can be integrated in an existing RedHat or Linux management and upgrade strategy.
- Low overhead from the installation itself. A Linux repository does not need much base CPU and Memory for the OS itself.
- A must have for certain ISO certifications and Cyber Insurance Policies.
- Reuse of existing hardware instead.
- Avoiding a vendor locked-in appliance.

Finally, it is also important to understand that a Veeam Hardened Repository is only a small puzzle in your complete cyber security. It might be a wise idea to educate your users, make sure that production applications are well protected, you have hardened network, etc. Cyber warfare is a constantly changing battlefield so it is also important to constantly evolve and update your strategy.