FIELD NOTES FROM A PRESALES SOLUTIONS ARCHITECT

Immutable Backups Ransomware Cannot Touch — What a Hardened Repository Really Requires

A presales field note about the word "immutable" on a backup proposal — the tick box that does nothing on its own — and the checks I now run before I tell anyone their backups survive ransomware.

Daanial Farrukh · 10 November 2026 · https://quabin.com/immutable-backups-hardened-repository-ransomware/

The requirement arrived as one line in a tender.

"The solution shall keep an immutable copy of all backups."

The customer already owned a large deduplication appliance. Their backup team was confident.

"We have immutability. It's in the job settings."

I asked one question.

"If an attacker gets your backup admin password tonight — can they delete last week's backups?"

A pause.

"Well… the admin can delete anything. That's what admin means."

That's the trap. Immutability does not show up as a setting you switch on. It shows up as an action that nobody can perform — not the attacker, not the backup software, and not you. If any account on the box can still remove the data before the lock date, you have a retention policy, not immutability.

What "Immutable" Actually Means

An immutable backup is a backup file that the storage itself refuses to change or delete until a set date, no matter which account asks.

The key word is *storage*. The backup software can ask for the lock. Only the storage can enforce it. The flow looks like this:

Backup server → transport service on the repository → file written → storage sets "no delete until date X"

Ransomware operators know this flow well. Their playbook is simple: get domain admin, find the backup console, delete the restore points, then encrypt production. So the real design question is not "is immutability ticked?" It is "what can a stolen admin password still do to the repository?"

Where I see the gap most often:

- **Windows repositories** with a backup folder on a domain-joined server — immutability ticked in the job, but any domain admin can format the disk.
- **Linux repositories** that are "hardened" in name only — SSH open, the backup user still in sudo, a stored root password in the backup server.
- **Deduplication appliances** where the retention lock was switched on from the web console — which, as you will see, can give you the weaker mode.

Why the Usual Check Misses It

The usual check is the backup console. It shows a padlock icon next to each restore point and a date. Everyone screenshots it for the audit.

That icon proves the software *asked* for the lock. It does not prove that nobody can walk around it. The two real tests are:

1. Try to delete a locked file on the storage itself, as the most powerful account you have.
2. List every way into the box and ask who holds each key.

Neither test happens in a normal go-live. Here is how I build and check both kinds of immutable storage now.

Path 1 — The Hardened Linux Repository

This is the design I run for a multi-tenant backup service: plain Linux VMs as repositories, an XFS file system, and Veeam's hardened repository mode with 7-day immutability. The software sets the immutable flag on each backup file; the Linux kernel enforces it, so even root cannot remove it until the date passes without first changing the flag — which is exactly what the lockdown below takes away.

Step 1: Build the file system for fast clone.

The repository disk gets XFS with reflink, so synthetic fulls are block clones instead of full copies:

```
mkfs.xfs -b size=4096 -m reflink=1,crc=1 /dev/vdb
```

Check it afterwards:

```
xfs_info /repo | grep reflink
```

You want reflink=1. If it says reflink=0, fast clone is off and every synthetic full will rewrite the whole chain.

Step 2: Give the backup folder to a plain user, not root.

```
chown repouser:repouser /repo/backups
chmod 700 /repo/backups
```

Step 3: Add the server with single-use credentials.

In the backup server, add the Linux host with the "single-use credentials for hardened repository" option and "elevate to root". The credentials are used once to deploy the transport service, then thrown away. Verify that the backup server really stores no SSH credential for this host. On ours it does not — which

means a stolen backup console cannot SSH into the repository.

Step 4: Firewall the box down to the data ports.

The repository only needs the transport port and the data range, and only from the backup network:

```
ufw default deny incoming
ufw allow from 10.0.0.0/24 to any port 6162 proto tcp
ufw allow from 10.0.0.0/24 to any port 2500:3300 proto tcp
```

Step 5: The lockdown — take your own keys away.

This step is where most tutorials stop and where mine starts. Until now, you could still log in over SSH as a user with sudo. So could anyone who stole that password. The lockdown removes every remote way in:

```
gpasswd -d repouser sudo
: > /root/.ssh/authorized_keys
ufw delete allow 22/tcp
systemctl disable --now ssh.socket ssh.service
```

Set a root password at the VM console before you run these. On our build, root had been locked since install. Removing sudo from the only admin user would have left GRUB recovery as the only way back in. After the lockdown the box is console-only, and console access sits with the hypervisor team, not with the backup admins.

Step 6: Verify from outside.

From the backup server:

```
nc -zv 10.0.0.47 22
nc -zv 10.0.0.47 6162
```

Port 22 must fail. Port 6162 must succeed. Then rescan the repository in the backup console; it must come back "Available". That combination — SSH refused, transport alive — is the proof that the repository works and that nobody can log in to it remotely.

Step 7: Check that the lock is really on a file.

At the console, after the first backup:

```
lsattr /repo/backups/<tenant>/<job>/*.vbk
```

You want an i in the attribute column. No i, no immutability — whatever the console icon says.

The Lockdown Is a One-Way Door — Plan for It

Two things surprised me after go-live, and both are the lock doing its job.

You will need SSH again, briefly. Re-adding the server to the backup software needs the single-use credentials, which need SSH. On one repository, two VMs on the same hypervisor both reported an all-zero BIOS UUID. The backup server merged them into one host record, and the repository went "Unavailable". The fix was to give each VM a unique UUID, open SSH for the backup server's IP only, remove and re-add the Linux server, and then lock down again. Write the re-lock into the change plan, because "lockdown still owed" is exactly the state an attacker hopes to find.

Check UUIDs before you add any VM-based repository:

```
cat /sys/class/dmi/id/product_uuid
```

If it is all zeros, or the same as another VM, fix it at the hypervisor first.

Immutable data cannot be moved or cleaned up early. When a tenant needed to move to a new repository, the move meant starting a new backup chain — the old chain stayed locked for its 7 days. When we built a test tenant, deleting it had to wait until its immutability ended. Put those dates in the plan before anyone promises a quick clean-up.

Path 2 — The Deduplication Appliance

The tender customer did not want new hardware. Their appliance (a Data Domain) supports retention lock in two modes, and the difference is the whole story.

- **Governance** — locked files can be unlocked by an administrator. It protects against mistakes, not against a stolen admin account.
- **Compliance** — nobody can unlock the files before the date, not even the system admin. Changes need a second person: a security officer.

Only Compliance answers the question I asked in the room.

The Trap: the Web Console Gives You Governance

On the appliance's web console, the "Modify Retention Lock" dialog has a toggle and min/max days — and no mode choice. We clicked OK on it once. The storage unit came out in **Governance** mode. Then we found the rule that hurts: a storage unit cannot be converted between Governance and Compliance. The CLI answered:

```
Operation to change retention-lock mode of mtree is not supported
```

The fix was a new storage unit, locked from the CLI, and the Governance one deleted. **Set Compliance from the CLI, on an empty storage unit, and nowhere else.**

The Order That Works

This is the order from the vendor's knowledge base, and the one that worked for us:

1. A security-officer account exists (only the first one can be made by sysadmin; after that only a security officer can add another).
2. As the security officer, enable the runtime authorization policy.
3. As sysadmin, system retention-lock compliance configure — the appliance reboots once.
4. Create iDRAC users from the appliance CLI. We did not plan for this. The enable command refused with "Create iDRAC users using DD OS interface before proceeding". Creating them disables the iDRAC root login and deletes the other iDRAC accounts, so agree this with the hardware owner first.
5. system retention-lock compliance enable — answer No to the date-change prompt.
6. Lock the storage unit in Compliance mode from the CLI:

```
mtree retention-lock enable mode compliance mtree /data/col1/<storage-unit>
```

Verify:

```
mtree list
```

The storage unit must show RLCE (Retention-Lock Compliance Enabled). If it shows RLGE, you have Governance and you are back to the trap.

Three Rules From the Backup Side

The backup software has its own rules for this kind of target. These three caught us:

- **Turn off automatic retention lock on the storage unit.** The backup software sets each file's lock date itself.
- **Your immutability period must sit inside the storage unit's min/max retention.** We kept the defaults (12 hours to 5 years) and used 30 days. A 30-day maximum on the storage unit would have blocked longer GFS points.
- **Backup copy jobs are only locked when GFS retention is on.** A copy job without GFS lands on the locked unit unlocked.

One more: the new backup server version ticks "Make recent backups immutable for 7 days" by default on a new repository. On a storage unit without Compliance, that tick fails with an error. We left the primary storage unit unlocked for fast daily jobs and put immutability on a second storage unit fed by a backup copy job with GFS. Two units, one locked — that's exactly the design.

The Checklist I Run Before I Write "Immutable"

1. **Is the lock enforced by the storage, not the software?** Linux: lsattr shows i. Appliance: mtree list shows RLCE.
2. **Can the strongest admin delete a locked file today?** Try it on a test file. The refusal is not a bug. It is the verification.
3. **Is SSH closed on the repository?** nc -zv <repo> 22 from the backup server fails.
4. **Does the backup server store any credential for the repository?** It should not.
5. **Who holds the console?** Name the person. It must not be the same account as the backup admin.
6. **Is the immutability period longer than the time an attacker sits inside before they strike?** 7 days is a floor, not a target.
7. **Are UUIDs unique on VM repositories?** cat /sys/class/dmi/id/product_uuid.
8. **Is the re-lock in every change plan that opens SSH?** Yes, written down, with a name next to it.

What I Tell Every Customer Now

I stopped asking "do you have immutable backups?" Everyone says yes.

I ask: "Show me the account that can delete last week's backup before its lock date."

If the answer is "there isn't one — we tried", they're in good shape. If the answer is "the admin, but only the admin", we're not done yet.

I write about the questions I ask in real customer rooms — the ones that change architectures and surface the problems no one else noticed. If you found this useful, drop your email below and I'll send

the next one when it's ready.

FAQs

Is ticking "make backups immutable" in the backup job enough?

No. The tick box asks the storage to lock the files. If the storage is a plain Windows folder, or a Linux box where an admin can still log in with sudo, or an appliance in Governance mode, a stolen admin account can still remove the backups. Immutability is real only when the storage refuses the delete for every account.

What is a hardened Linux backup repository?

It is a Linux server that stores backups with the immutable file flag set by the backup software, with no remote admin access left. It is added with single-use credentials, firewalled down to the transport and data ports, and then locked down: no sudo for the service user, no SSH keys, SSH disabled, console-only access.

What is the difference between Governance and Compliance retention lock?

Governance lets an administrator unlock files before their date, so it protects against mistakes, not against a stolen admin account. Compliance lets nobody unlock files early, and changes need a separate security-officer account. A storage unit cannot be converted from one mode to the other, so set Compliance from the CLI on an empty unit.

How do I check that a backup file is really immutable?

On a hardened Linux repository, run lsattr on the backup files at the console and look for the i attribute. On a deduplication appliance, run mtree list and look for RLCE on the storage unit. Then try to delete a test file as the strongest account you have; the refusal is the proof.

How long should the immutability period be?

At least 7 days, and longer than the time an attacker is likely to sit inside your network before they strike. Keep it inside the storage unit's min and max retention on an appliance, and remember that locked data cannot be moved or deleted early, so plan migrations and clean-ups around the lock date.

Read online: https://quabin.com/immutable-backups-hardened-repository-ransomware/