WHITE PAPER · ENTERPRISE BACKUP SECURITY

SECURITY CHECKLIST

Self-Hosting Bacula: The Complete Security & Operations Checklist

34 checkpoints across 12 domains. Installing Bacula is the easy part.

A structured checklist for teams deploying and operating Bacula themselves. Covers OS hardening, access control, encryption, monitoring, restore testing, and disaster recovery — the decisions that separate a functioning backup installation from a resilient one.

OS hardening, firewall, access control, encryption, GUI security, scripting review, logging, restore testing, and more.

PREPARED BY
faaleoleo

PLATFORM
Bacula Enterprise

VERSION
1.0

CONTENTS

Executive Summary

Installing Bacula is not the hard part. A working installation can be up in an afternoon. What takes months to get right — and what is almost always incomplete in self-hosted environments — is everything that makes that installation secure, observable, and recoverable.

WHAT THIS CHECKLIST COVERS

- **Foundation:** Operating system selection, minimal installation, firewall policy, and host-based intrusion detection — the baseline every production server must meet before Bacula is even installed.
- **Access control:** User privilege separation, restricted shells for backup operators, two-factor authentication at the OS level, on the management GUI, and across all administrative interfaces.
- **Bacula-specific security:** Per-client passwords, TLS transport, client-side encryption, Storage Daemon volume protection, configuration quality, and scripting hygiene.
- **Operations:** Logging, compliance reporting, audit trails, backup scope verification, and the automated restore testing that separates a real backup from a hypothesis.
- **Resilience:** Catalog database backup, bootstrap file retention, database maintenance, documentation completeness, and a tested disaster recovery plan.

34	12	1	0
CHECKPOINTS	SECURITY DOMAINS	GOAL: NO GAPS	SHORTCUTS

Each checklist item includes a brief explanation of why it matters. Use this document to assess an existing installation, plan a new deployment, or understand what a complete Bacula operation actually requires.

The checklist is organised by domain, moving from the server foundation outward to application-level controls, and finally to operational and continuity requirements. All items apply to every production Bacula installation — none are optional.

1. Operating System

> The operating system is the foundation of your backup infrastructure. Security decisions made — or not made — at this level affect every component above it.

1 Operating System Selection & Maintenance

- [ ] **You are running a current, actively maintained server Linux distribution**

  Ubuntu LTS, Debian stable, RHEL/AlmaLinux/Rocky Linux, and SLES are common production choices. Check the vendor's end-of-life schedule. An EOL operating system receives no security patches — every CVE published after that date is permanently unaddressed on your backup server.

- [ ] **Your installation is minimal — no GUI, no development tools, no services you do not need**

  Every installed package is a potential attack surface. A backup server has one job. Strip the OS to what that job requires and nothing more. Disable or remove network services that are not needed: FTP, Telnet, CUPS, Avahi, and any others that appear in `ss -tlnp` without a reason to be there.

- [ ] **Security updates are applied promptly and automatically where safe**

  Unattended-upgrades (Debian/Ubuntu) or dnf-automatic (RHEL) handle security patches without manual intervention. Test that the update mechanism is actually running: a broken update pipeline provides false confidence.

2. Network & Firewall

A firewall that only blocks inbound traffic is half a firewall. Outbound filtering limits the blast radius of a compromised process and detects software attempting to call home without your knowledge.

2 Network & Firewall Policy

- [ ] **Your firewall controls both inbound AND outbound traffic**

  Default-deny outbound with explicit allowlists for required destinations is the correct posture. A backup server needs outbound access to: the configured mail relay (port 587 or 465), package repositories (80/443), and nothing else. Everything else should be blocked and logged.

- [ ] **Bacula network ports are restricted to known source hosts only**

  Director (9101), File Daemon (9102), and Storage Daemon (9103) must not be reachable from the public internet. Restrict access to the specific IP addresses of your Bacula infrastructure. If clients exist outside your network perimeter, use a VPN rather than exposing Bacula ports directly.

- [ ] **SSH access is locked down — key-only authentication, non-standard port or port-knocking where appropriate**

  Password authentication for SSH should be disabled. `PasswordAuthentication no` in `sshd_config` is the minimum. Consider AllowUsers/AllowGroups to limit which accounts can use SSH at all.

3. User Management & Access Control

> Access control failures are the most common vector for backup infrastructure compromise. Overprivileged accounts, shared credentials, and missing multi-factor authentication each represent a single point of failure.

3 User Privileges & Authentication

- [ ] **Direct root login is disabled — administrative tasks require `sudo`**

  `PermitRootLogin no` in `sshd_config` and `root` shell set to `/usr/sbin/nologin`. Root access is the nuclear option. `sudo` gives you an audit trail, the ability to revoke privileges without changing passwords, and the ability to grant specific commands without full root. There is no legitimate reason for interactive root login on a production server.

- [ ] **All user accounts apply the principle of least privilege**

  A user who only needs to monitor backup job status does not need write access to the Bacula configuration, the ability to run restores, or access to the storage volumes. Review `/etc/sudoers` and remove any `NOPASSWD: ALL` grants that are not strictly justified.

- [ ] **System administrators and Bacula operators are separated — operators use a restricted shell**

  A Bacula operator role — someone who monitors jobs, initiates restores, and manages schedules — does not require system administrator capabilities. `rbash` or a custom shell wrapper restricts what a logged-in session can do. This limits the damage from a compromised operator account to the Bacula application layer, not the entire host.

- [ ] **Two-factor authentication is enforced for all accounts with SSH or console access**

  A stolen SSH key or cracked password alone must not be sufficient to authenticate. TOTP (via `libpam-google-authenticator` or `pam_oath`), FIDO2/WebAuthn hardware tokens, or SSH certificate authority with short-lived certificates all provide a second factor. Choose one and enforce it across all privileged accounts without exception.

4. Intrusion Detection

> A host-based intrusion detection system (HIDS) is the difference between discovering a breach during the attack and discovering it after the fact — or not at all.

4 Host-Based Intrusion Detection

- [ ] **A host-based IDS is running and actively enforced**

  CrowdSec is the recommended modern choice — it aggregates threat intelligence across deployments and provides automated IP blocklists. **fail2ban** is the widely deployed alternative, monitoring log files for patterns (brute-force SSH, repeated authentication failures) and issuing firewall bans. Either must be configured to monitor `/var/log/auth.log`, Bacula's own logs, and your web server logs if the management GUI is exposed. Alerts should notify on ban events.

- [ ] **File integrity monitoring is in place for critical system paths**

  AIDE, Tripwire, or OSSEC can detect modifications to system binaries, configuration files, and the Bacula configuration directory. Run baseline checks after known-good installations and alert on unexpected changes. A changed Bacula binary or Director configuration is a significant security event.

5. Alerting & Monitoring

> Monitoring that nobody reads and alerting that does not work are worse than nothing — they create a false sense of coverage while providing no actual protection.

5 Alerting & Operational Monitoring

- [ ] **Email alerting is functional, tested, and uses TLS — relay is restricted to localhost**

  Your mail setup must accept relay only from `127.0.0.1` (no open relay). Outbound mail must use STARTTLS or SMTPS to the configured smarthost. Test it: send a test alert and verify delivery end-to-end. Many installations have broken alerting that nobody notices until a critical backup fails silently for days.

- [ ] **System monitoring is in place and alerts are understood and acted upon**

  CPU, disk, memory, and service health monitoring must cover the Director, Storage Daemon, and catalog database. Alerts must reach someone who understands what they mean and knows how to respond. A monitoring system that generates noise until it is muted is not a monitoring system — it is false confidence.

- [ ] **Every day, you know whether all backup jobs completed successfully — without having to look manually**

  A daily summary report must land in someone's inbox — or a dashboard must show green — before the business day begins. If the only way to know that last night's backups succeeded is to log in and check, then on the days nobody checks, you have no coverage. Bacula's `Messages` resource can send consolidated job reports by email. Pair this with a dead-man's-switch alert: if no report arrives by a certain time, that itself triggers a notification. Silence should be an alarm, not reassurance.

6. Backup Management Interface

Web-based management interfaces are high-value targets. The access controls protecting the Bacula GUI must be at least as strong as those protecting SSH — ideally stronger, because the attack surface is larger.

6 GUI Access Security

- [ ] **The management GUI is not accessible from the public internet**

  Bacularis, BWeb, and bacula-web must be protected by network-level access controls: firewall rules restricting access to specific source IPs or subnets, a VPN requirement, or an nginx/Apache ACL. "I use HTTPS" is not a substitute for network restriction. A login page exposed to the internet is a login page being attacked continuously.

- [ ] **GUI users have scoped, role-based access — there is no shared admin account**

  Individual named accounts with role-based permissions enable you to see who ran a restore, who modified a job schedule, and who disabled an alert. A shared admin account with a team password makes this reconstruction impossible and makes credential rotation operationally difficult. Every user who needs GUI access gets their own account with the minimum role required.

- [ ] **Two-factor authentication is enforced for all GUI accounts**

  TOTP, FIDO2, or integration with an identity provider that enforces 2FA (OIDC/SAML) must be required for GUI login without exception. The backup management interface provides the ability to initiate restores, cancel jobs, and modify what gets backed up. Access to it must require more than a password.

7. Bacula Configuration & Scripting

A Bacula configuration built by trial and error will have gaps. FileSets with incomplete exclusions, schedules that produce unexpected overlaps, retention policies that were never verified — these are the normal outcome of informal configuration approaches. Scripts are a more serious problem: a script that does its job but is insecure is a vulnerability, not a solution.

7 Configuration Quality & Scripting Hygiene

- [ ] **Your Bacula configuration follows documented best practices — not ad-hoc trial and error**

  Configuration validation (`bacula-dir -t` , `bacula-sd -t` , `bacula-fd -t`) verifies syntax, not correctness. A configuration that passes validation can still have logic errors that only surface during a recovery. Follow the official Bacula documentation for FileSets, pool configuration, retention hierarchies, and concurrent job limits. Treat configuration changes as code changes: review, test, commit.

- [ ] **Custom scripts have been reviewed for both correctness and security**

  Pre-backup hooks, database dump scripts, and notification handlers are common in Bacula installations. Every script must be reviewed before deployment. The following example illustrates a script that may appear to work but is insecure in multiple ways:

  Security Issues — Do Not Use This Pattern:

  ```
  #!/bin/bash
  mysqldump -u root -ppassword123 --all-databases > /tmp/dump.sql
  ```

  Problem 1 — Hardcoded credentials: The password `password123` is visible to any user who can run `ps aux` while the script executes, is stored in shell history, and is readable by anyone with access to the script file.

  Problem 2 — Root database access: A dedicated backup user with `SELECT` and `LOCK TABLES` on the required databases is all that is needed. Running as the MySQL root user gives the script — and anything that exploits it — full database control.

  Problem 3 — World-readable output: `/tmp` is readable by all local users. A database dump written there is readable by every process and every user on the system until it is deleted. Use a directory owned by the backup service user with mode 0700.

  Problem 4 — No error handling: If `mysqldump` fails silently, the script exits cleanly and Bacula backs up an empty or partial file. Include error checking and verification of the output.

- [ ] **Each Bacula client (File Daemon) has its own unique password**

  A shared password across multiple File Daemons means a single compromised client can authenticate as any client. The Director trusts passwords, not identities. Unique per-client passwords ensure that a compromise is contained to a single node. TLS with client certificates provides stronger guarantees and should be used wherever possible.

8. Encryption

Encryption must be addressed at three independent layers: data at rest on the client before transmission, data in transit between Bacula components, and data at rest on the Storage Daemon. Each layer is a separate control. Missing any one of them leaves a gap.

8 Encryption — Three Independent Layers

- [ ] **Client-side data encryption is enabled — data is encrypted before leaving the client**

  Bacula Enterprise supports PKI-based client-side encryption using per-client certificate and key pairs. Data is encrypted on the File Daemon before transmission and remains encrypted at rest on the Storage Daemon. Even with full access to the storage volumes, the data is not readable without the client key. Keys must be stored independently of the backup infrastructure itself.

- [ ] **TLS is configured for all connections between Bacula components**

  Director ↔ File Daemon and Director ↔ Storage Daemon communication must use TLS. Without it, job metadata, file paths, and potentially file contents travel in clear text across your network. Configure `TLS Enable = yes`, `TLS Require = yes`, and provide certificate and CA paths in all Bacula resource definitions. Verify with a packet capture that the handshake succeeds.

- [ ] **Storage Daemon volumes are encrypted at rest and protected against tampering**

  Full-disk encryption (LUKS) or filesystem-level encryption on the storage volumes ensures that physical or remote access to the storage medium does not produce readable data. Tamper protection — checksums or volume signatures verified by Bacula — detects modifications made outside of normal backup operations. Both controls are needed: encryption for confidentiality, integrity verification for detection.

- [ ] **You understand FIPS 140-2/3 and whether it applies to your environment**

  FIPS (Federal Information Processing Standard) 140-2 and its successor 140-3 are US federal standards specifying approved cryptographic algorithms and implementations. They are required for US government contractors, healthcare organisations under certain regulations, and financial institutions subject to specific frameworks. If you are in scope, your Linux distribution's FIPS-validated kernel module (`fips=1`) and OpenSSL FIPS provider must be enabled — not just standard encryption. If you are unsure whether FIPS applies to your organisation, consult your compliance team before deployment.

9. Logging, Reporting & Audit

> Compliance frameworks do not ask whether backups ran. They ask whether you can demonstrate that they ran, what was backed up, and who had access. Logging and reporting are what make that demonstration possible.

9 Logging, Reporting & Audit Trails

- [ ] **Bacula job logging is configured with rotation and long-term retention**

  Default Bacula logging often writes to a flat file with no rotation. On a busy Director, this file grows without bound. Configure `logrotate` for Bacula's log file and forward logs to a central syslog server (`rsyslog` or `syslog-ng`). Central logging means log data survives a host compromise and is available for forensic analysis. Retention must match your compliance requirements — typically 90 days minimum, often 1–3 years.

- [ ] **You receive and review compliance reports demonstrating backup coverage**

  Can you produce a report showing that every protected system was backed up within its required window, how much data was transferred, and which jobs failed or were skipped? If not, you cannot demonstrate operational compliance even if your data is technically safe. Bacula's built-in reporting via `bconsole` commands (`list jobs`, `status director`) can be scripted and scheduled. Management reports must be reviewed — not just generated.

- [ ] **Audit logs exist — you can determine who did what, and when**

  bconsole commands, REST API calls, and GUI logins must produce an audit trail. This is a hard requirement for PCI-DSS, ISO 27001, NIS2, SOC 2, and most other frameworks. Audit logs must be tamper-evident — ideally written to a remote log collector that the backup server cannot modify. Verify that your audit log retention policy covers the periods required by your applicable frameworks.

10. Backup Scope & Completeness

What you do not back up does not exist in a recovery scenario. Backup scope drift — new systems added without corresponding backup jobs, databases deployed outside the backup perimeter, new cloud storage never added to FileSets — is the most common cause of data loss in organisations that believed they were protected.

10 Backup Scope & Coverage Verification

- [ ] **Your backup scope is accurate — everything that must be protected is included**

  Maintain a formal backup scope register: every system, database, and service that requires protection, mapped to the specific Bacula job that protects it. Review the register against your actual infrastructure inventory quarterly — or whenever a new system is provisioned. New servers, new databases, new cloud-mounted storage, and new applications all need explicit scope decisions.

- [ ] **FileSets are precise — exclusions are intentional and nothing critical is silently skipped**

  Overly broad FileSet exclusions are a frequent source of data loss. Exclusions like `Exclude { File = /var }` may skip database files, application state, or logs that are critical to recovery. Review every exclusion rule and verify that the files being excluded are genuinely not needed for recovery. If in doubt, include and manage volume growth with retention policies rather than exclusions.

11. Testing & Recovery

A backup that has never been restored is a hypothesis. Automated restore testing is the only reliable way to know that your backups are usable. A disaster recovery plan that has never been exercised is a document, not a capability.

11 Restore Testing & Disaster Recovery

- [ ] **Automated restore tests run regularly and results are reviewed**

  Schedule restore verification jobs that pull recent backup data to a test environment and verify integrity. The result must be reviewed — not just logged. A restore test that succeeds but is never checked provides the same operational assurance as no restore test at all. Run restore tests at least monthly for critical systems; weekly for the most critical.

- [ ] **A tested disaster recovery plan exists and is documented**

  "I know how to restore things" is not a disaster recovery plan. A tested plan specifies: the recovery time objective (RTO) for each system, the recovery point objective (RPO), the exact steps to rebuild the Bacula infrastructure from scratch, how to restore the catalog database, and the order of system restoration for dependency-aware recovery. It must have been walked through at least once with a team member who was not the author.

12. Database & Infrastructure

> The Bacula catalog database is not just application data — it is the index to every backup Bacula has ever performed. Losing it without a backup means losing the ability to perform targeted restores, even if the backup data itself is intact. It must be treated with the same care as any other critical database.

12 Catalog Database & Operational Infrastructure

- [ ] **The Bacula catalog database is backed up independently of Bacula**

  The catalog must be exported via `mysqldump` or `pg_dump` on a schedule that matches your RPO, and stored somewhere other than the Bacula infrastructure itself. Without the catalog, restore operations are limited to volume-based recovery using bootstrap files — partial at best. The catalog backup must be tested: a corrupt or incomplete dump discovered during a recovery is not a backup.

- [ ] **Bootstrap (.bsr) files are retained and stored in a location accessible without the catalog**

  Bootstrap files allow volume-based recovery when the catalog is unavailable. They must be stored outside the Bacula Director — on a separate system, in offline storage, or in a location that survives a Director failure. A .bsr file is your last resort; keeping it only on the Director eliminates the scenario in which it matters most.

- [ ] **The catalog database is installed and configured per best practices — not default settings**

  Default MySQL/PostgreSQL installations often use socket-only authentication with no password for the root account, generic character sets, and no performance tuning. Bacula's catalog schema has specific requirements. Use Bacula's provided `create_bacula_database`, `make_bacula_tables`, and `grant_bacula_privileges` scripts rather than manual database setup. Configure a dedicated database user for Bacula with only the permissions those scripts grant.

- [ ] **Database maintenance is scheduled — pruning, vacuuming, and reindexing run regularly**

  The Bacula catalog grows continuously. Without regular pruning (via `bconsole prune` commands or automated Bacula pruning rules), query performance degrades, disk usage grows without bound, and eventually backup jobs begin timing out because catalog queries become too slow. For PostgreSQL installations, autovacuum must be configured and monitored. Run `ANALYZE` after large prune operations.

- [ ] **Your Bacula infrastructure runs on dedicated resources — not shared with other workloads**

  A backup server competing for CPU, memory, I/O, or network bandwidth with production workloads is a backup server that will underperform precisely when it is under the most pressure — during large backup windows, during recovery operations, or during incidents when both backup and production systems are heavily used simultaneously. The Director, Storage Daemon, and catalog database should each have guaranteed, predictable resources. In virtualised environments this means reserved CPU and memory, not shared pools. Resource contention during a recovery is not a problem you want to solve under time pressure.

13. Documentation & Continuity

An undocumented configuration is a configuration that only one person understands. When that person is unavailable — on leave, sick, or no longer with the organisation — the backup infrastructure becomes opaque. Decisions made without documentation cannot be reviewed, audited, or safely changed.

13 Documentation & Knowledge Continuity

- [ ] **The complete Bacula configuration is documented — decisions made months ago are explainable**

  Documentation must answer: why does this FileSet exclude this path? Why is this job scheduled at this time rather than another? Why does this pool have this retention period? The configuration files themselves are not documentation — they describe what, not why. Version-control the configuration and use commit messages to capture rationale. Supplement with a runbook that explains non-obvious decisions and any known limitations of the current setup.

- [ ] **Audit logs are retained in accordance with applicable compliance framework requirements**

  Audit log retention requirements vary significantly by framework: NIS2 typically requires 12 months, PCI-DSS requires 12 months with 3 months immediately available, ISO 27001 is policy-defined but typically 12–24 months. Verify that your log retention settings, rotation policies, and archive procedures produce the required retention period. Test the archive retrieval process — logs that exist but cannot be retrieved are not usable for audit purposes.

Conclusion

This checklist has 32 items. That number is not meant to be intimidating — it is meant to be accurate. A production-ready, secure, and operationally sound Bacula deployment requires deliberate attention across all of these domains.

Most self-hosted Bacula installations satisfy some of these requirements well. The OS is current. The firewall has inbound rules. Backups run and jobs complete. But gaps in the less visible areas — outbound firewall policy, operator privilege separation, scripting hygiene, automated restore testing, catalog database backup, audit log retention — are the norm, not the exception.

The goal of this document is to make those gaps visible before a security incident or a recovery scenario makes them consequential.

> **A working backup is not the same as a recoverable backup.** Job completion status confirms that data was written to storage. It does not confirm that the data is readable, that the catalog accurately reflects what was written, that the restore process will work under time pressure, or that the team performing the recovery knows what to do. Only tested restores and practised recovery procedures confirm those things.

Use this checklist to identify the gaps in your current installation. Address them systematically, in priority order: access control and encryption first, then operational controls, then documentation and continuity. Revisit the checklist quarterly — infrastructure changes, people change, and compliance requirements evolve.