Known Limitations, Troubleshooting & Version History

Known Limitations & Accepted Risks

LXC auditd compliance gap

Affected hosts: `pbr-graylog-kl1`, `pbr-thingsboard-kl1`

Issue: auditd cannot run inside LXC containers. The kernel audit netlink interface is isolated from container namespaces. Forcing auditd to start would fail with EPERM at the systemd start.

v2.4.2 introduced auto-detection: hosts with `ansible_virtualization_type == 'lxc'` have auditd installation but no service enablement. The `verify.yml` auditd assertion is skipped on these hosts.

Compliance implication: No local audit log capture on those two hosts. Compliance evidence for them depends entirely on remote logging via Graylog SIEM (system journal forwarding, application-level logs).

Mitigations in place:

- Both LXC hosts forward system events to Graylog
- Both run a limited service set with constrained external exposure
- Operating system logs are still captured via journald and forwarded

Future options to close the gap:

1. Migrate the affected workloads to KVM VMs (decouples from container constraints, restores local audit log capture)

2. Investigate Proxmox VE 9's enhanced container support for the audit subsystem (may not be available)
3. Formally accept the residual risk in PBR's risk register, citing the SIEM-based compensating control

---

Realm join multi-master replication retry pattern

Observed: During the v2.4.2 rollout, 3 of 5 hosts needed two attempts to complete `realm join` despite proper AD pre-clean.

Root cause: AD multi-master replication lag across PBR's 4 DCs. The `realm join` command picks a DC (via SRV record lookup), but that DC may not have replicated the deletion of the previously-cleaned-up computer object yet. The join then fails because "the object already exists."

Mitigation: Re-run the playbook. The role is idempotent, and by the time the second attempt runs, replication has usually caught up. The second attempt almost always succeeds.

Why we haven't added automatic retries: A `retries: 2, delay: 30` on the join task would mask the behaviour from operators. While that's convenient, it also hides a real symptom that's worth observing. Deferred to v2.5 with the intent to add retries plus a debug message about the replication-lag pattern.

ThreatLocker sudoers permission issue

Observed on: All hosts with ThreatLocker installed.

Issue: ThreatLocker's agent installs `/etc/sudoers.d/threatlocker_sudoers_general` with incorrect permissions. The file should be mode 0440 but is set to something `visudo -c` rejects. ThreatLocker enforces file immutability on its own files, so the permissions cannot be corrected.

Effect: `sudo` on the host does not honour the contents of that drop-in (it's rejected during sudoers parsing). Whatever rules ThreatLocker intended to install via that file are inactive.

Workaround in the role: preflight's `visudo -c` task ignores stderr lines mentioning `threatlocker_sudoers_general`. Any other sudoers error still fails preflight.

Action item: Raise with ThreatLocker support. Preflight emits a clear debug message when the workaround fires, so the operator is reminded each run.

Royal TS Rebex SSH library cannot do AuthenticationMethods publickey,keyboard-interactive

Issue: Royal TS 7's bundled Rebex SSH library does not support OpenSSH's `AuthenticationMethods publickey,keyboard-interactive` directive natively — it only handles one authentication method per session.

Symptoms: Royal TS fails to connect to baselined hosts with errors about authentication negotiation, or completes publickey auth and then disconnects without prompting for Duo.

Workaround: Set Royal TS's authentication method to `Any` under the connection's **Advanced → Security** properties. This lets Rebex negotiate either method, and the server-side `AuthenticationMethods` directive still requires both. The Duo keyboard-interactive prompt is then handled by the connection's interactive shell.

Alternative: Configure Royal TS to launch Windows OpenSSH (`ssh.exe`) as an External Application connection. Native OpenSSH handles `AuthenticationMethods` correctly and integrates with the 1Password SSH agent via the named pipe.

Hardcoded bootstrap SSH public key

Observed in: `scripts/bootstrap-ansible-user.sh`

The bootstrap script contains the control node's public key as a string literal:

```
PUBKEY="ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBMc7IDlr/IZ5M/2HcXU7cGCKZ03SLjpr5cbmiHnokdP
ansible-svc@pbr-ansible-kl1"
```

If the control node is rebuilt with a new ed25519 keypair, this script must be updated. The provenance comment in the script's banner explains the source.

This is a known trade-off: the script must work in isolation (run on a fresh host before any Ansible config is in place), so a hardcoded key is simplest. The alternative — templating the key into the script — would require a different deployment mechanism for the bootstrap step.

Banner file (issue.net) source not currently in repo dump

The role deploys `/etc/issue.net` from `roles/ssh-baseline/files/issue.net` via the `Deploy SSH login banner` task in `sshd.yml`. The banner file itself was not present in the v2.4.2 code dump used to author this documentation. To inspect the live banner, check any baselined host:

```
cat /etc/issue.net
```

Troubleshooting Reference

"User <user> from <ip> not allowed because none of user's groups are listed in AllowGroups"

Symptom: SSH connection rejected before authentication. Visible in the client with `ssh -vvv` and in `journalctl -u ssh` on the host.

Cause: The user is not a member of any group listed in sshd's `AllowGroups` directive (`sudo`, `sg_serveraccess`, `sg_sudo`).

For local accounts (ansible, pbr_admin): Verify membership in the local `sudo` group:

```
id ansible | tr ',' '\n' | grep -i sudo
id pbr_admin | tr ',' '\n' | grep -i sudo
```

If `ansible` isn't in `sudo`, re-run the role — v2.4.1's `preconditions.yml` adds it idempotently. This was the v2.4 → v2.4.1 fix.

For AD users: Verify SSSD resolves their group membership:

```
id a.mfraser
# Expected: a member of sg_serveraccess and/or sg_sudo (lowercased)
```

If the AD group memberships don't show, SSSD cache may be stale: `sudo sss_cache -E`.

realm join fails with no_log censored output

Symptom:

```
TASK [ssh-baseline : Join Active Directory domain] ****************************
fatal: [pbr-NEWHOST-kl1]: FAILED! =&gt; changed=true
  censored: 'the output has been hidden due to the fact that no_log: true was specified for
this result'
```

Most common cause: AD multi-master replication lag (the host being joined hits a DC that hasn't seen the previous computer object's deletion). Fix: re-run the playbook.

If second attempt also fails, dig deeper:

```
ansible pbr-NEWHOST-kl1 -m shell -a '
    journalctl --since "10 minutes ago" --no-pager 2&gt;&amp;1 \
        | grep -iE "realm|adcli|krb5|sssd|kerberos" | tail -40
    timedatectl status | head -8
' --become --vault-password-file ~/.ansible_vault_pass
```

Look for: clock skew (Kerberos requires <5 min skew with KDC), DNS resolution failures, computer object already exists messages, "krbtgt" related errors (KDC contact failures).

Last resort — temporarily remove no_log: Edit `roles/ssh-baseline/tasks/ad-join.yml`, comment out the `no_log: true` on the realm join task, re-run with output going to stdout (not `tee`'d to disk). Restore `no_log: true` immediately after. Scrub any tee'd diagnostic logs with `shred -u`.

SSSD user doesn't resolve via getent

```
getent passwd a.mfraser
# (no output, rc=2)
```

Possible causes (test in order):

1. **User not in SG_ServerAccess or SG_Sudo** — The `ad_access_filter` in SSSD excludes them. Check group membership in ADUC.

2. **SSSD service not running** — `systemctl status sssd`. If down, `systemctl start sssd` and check journalctl for the failure reason.
3. **SSSD cache stale** — `sudo sss_cache -E` invalidates the cache; SSSD re-queries AD on next lookup.
4. **SSSD offline** — `sssctl domain-status pbr.org.au`. ONLINE means LDAP is reachable; OFFLINE means SSSD has lost contact with DCs.
5. **LDAP connectivity broken** — verify DC reachability: `nc -zv 10.1.8.90 389; nc -zv 10.1.8.90 88`.

SSH key not retrieved from AD

Symptom: sshd publickey auth fails for an AD user whose `sshPublicKey` attribute is populated.

Diagnostic: Run the same lookup sshd does:

```
sudo -u nobody /usr/bin/sss_ssh_authorizedkeys a.mfraser
```

Expected: The user's public key on stdout.

If empty:

- Verify `sshPublicKey` populated on the AD user object (in ADUC or via PowerShell `Get-ADUser a.mfraser -Properties sshPublicKey`)
- Verify SSSD's ssh responder is running: `services = nss, pam, ssh` in `/etc/sssd/sssd.conf`. Re-deploy the role to restore if drifted.
- Verify SSSD is online: `sssctl domain-status pbr.org.au`
- Clear cache: `sudo sss_cache -E` and retry

Duo: "Permission denied" without a Duo prompt

Cause: Auth rejected before PAM ran. Most likely `AllowGroups` rejected the user.

```
ssh -vvv a.mfraser@host.pbr.org.au 2&gt;&amp;1 | grep -iE 'permission
denied|allowgroups|publickey'
```

Also possible: publickey auth failed (no matching key in AD) and the connection terminated before keyboard-interactive.

Duo: prompt arrives but authentication fails

Check the host's Duo logs:

```
sudo journalctl -u ssh --since "5 minutes ago" | grep -iE 'duo|pam'
```

Common causes:

- Duo API credentials wrong in `/etc/duo/pam_duo.conf` — vault credentials mismatch. Re-run the role to refresh.
- System clock drift — Duo's API requires close NTP sync. `timedatectl status`.
- User disabled in Duo admin console.
- User's primary device unreachable (no network on phone, app not installed).

---

sudo asks for password but never prompts for Duo

Cause: User is not in `sg_sudo`, so the `pam_succeed_if user notingroup sg_sudo` branch fired and skipped pam_duo. By design.

```
id a.mfraser | tr ',' '\n' | grep -i sg_sudo
```

If the user should be in sg_sudo but isn't showing: stale SSSD cache. `sudo sss_cache -E`.

Local sudo broken after role run

Caught by the role itself — the validation task `Sanity check - sudo still works for non-Duo automation accounts` runs `sudo -n true` as the `ansible` user during deployment. If this fails, the playbook aborts with a clear error, before later tasks that depend on working sudo.

If it does break (e.g. a manual edit to `/etc/pam.d/sudo` went wrong):

```
# As pbr_admin (break-glass, password auth):
ssh pbr_admin@&lt;host&gt;
sudo -i
```

```
# Restore Ubuntu default:
DEBIAN_FRONTEND=noninteractive apt-get install --reinstall -y \
    -o Dpkg::Options::="--force-confmiss" sudo

# Then re-run the role to restore the Duo-aware /etc/pam.d/sudo properly
```

SSH times out from one source only — fail2ban ban (incl. self-lockout)

Symptom: SSH to a baselined host *times out* — not "Connection refused", not "Permission denied" — and only from your source IP, while the host is up and other sources can still connect. fail2ban's drop action discards packets silently, so a ban presents as a connection *timeout* rather than a refusal. Repeated failed logins (or repeated attempts that reached the auth stage) can ban your own admin workstation.

Confirm the ban — needs another route onto the box (Proxmox/iKVM console, an unbanned source, or the Ansible control node, which is in `ignoreip`):

```
fail2ban-client status                # list active jails
fail2ban-client status sshd           # jail detail: currently banned + Banned IP list
fail2ban-client get sshd banned       # banned IPs for the sshd jail (fail2ban 0.11+)
grep "Ban " /var/log/fail2ban.log     # ban history  (or: journalctl -u fail2ban)
```

Release a ban:

```
fail2ban-client set sshd unbanip <ip>   # unban a specific IP in the sshd jail (always
available)
fail2ban-client unban <ip>              # unban an IP across all jails (fail2ban 0.10+)
fail2ban-client unban --all           # clear every ban in every jail
```

Prevent recurrence: add the trusted admin/control source to the fail2ban `ignoreip` allow-list in `roles/ssh-baseline/defaults/main.yml` (the same list that already exempts `127.0.0.1/8`, `::1`, the server LAN, and the `10.1.8.80/32` control node). IPs in `ignoreip` are never banned. Note POS hosts override `fail2ban_sshd_bantime` to `604800` (7 days), so a self-ban there persists far longer than the default.

Version History

2026-06-17 — pbr_admin source allow-list widened to 10.0.0.0/8 (role version tag TBC)

- Changed default `pbr_admin_allowed_sources` from `10.1.0.0/16,192.168.0.0/16` to `10.0.0.0/8,192.168.0.0/16`. The explicit `10.1.0.0/16` was dropped as it is a subset of `10.0.0.0/8`; `192.168.0.0/16` (TEMPORARY) retained.
- Effect: the `pbr_admin` break-glass Match block now accepts password auth from any host on the internal 10/8 estate, not just the server LAN. Per-host overrides (e.g. `pbr-pos-belgrave.yml` pinned to `10.1.8.0/24`) are unaffected.
- Trade-off noted: `pbr_admin` is password-only (carved out of the Duo PAM flow), so this widens the source scope of the one MFA-exempt account. Accepted as an interim measure; revisit narrowing to per-site management CIDRs once VLAN segmentation consolidates admin sources (tracked with the `192.168.0.0/16` TEMPORARY entry).
- Docs updated: SSH Hardening Reference (Match block + rendered source + scope note) and Configuration Reference (variable default). Role version tag to be assigned on commit.

v2.4.2 (current)

Title: Auto-skip auditd on LXC containers

Commit: `6286698` (with companion commits `296ab08`, `52befaf`, `56c0f73`)

Changes:

- `packages.yml`: added `set_fact: _manage_auditd` with auto-detection logic. Conditional `Enable auditd` service task.
- `verify.yml`: duplicate auto-detection added so verify works independently of packages.yml. Auditd assertion gated on `_manage_auditd`.
- `defaults/main.yml`: `manage_auditd: auto` default with explanatory comment.
- Companion: `scripts/bootstrap-ansible-user.sh` added to the repo (was previously informal).
- Companion: `296ab08` restored `no_log: true` on the realm join task (a temporary removal during diagnostic work).
- Companion: `52befaf` added `pbr-thingsboard-kl1` to inventory.

Rolled out: All 5 hosts — `pbr-uisp-kl1`, `pbr-docker-kl1`, `pbr-graylog-kl1`, `pbr-lme-kl1`, `pbr-thingsboard-kl1`.

---

v2.4.1

Title: Ensure ansible automation account is in sudo group

Commit: `4eb86b4`

Problem: After v2.4's `AllowGroups sudo sg_serveraccess sg_sudo` took effect on hosts where the `ansible` account had been bootstrapped historically without sudo group membership, sshd rejected the ansible connection with "User not allowed because none of user's groups are listed in AllowGroups."

Why it surfaced: The canary host (`pbr-uisp-kl1`) had had `ansible` added to `sudo` by an earlier manual bootstrap. `pbr-docker-kl1` did not. When v2.4 rolled to docker-kl1 with the hardened `AllowGroups`, the ansible session was severed mid-deployment.

Fix: `preconditions.yml` now runs as the first task of the role:

```
- name: Ensure ansible automation account is in local sudo group
  ansible.builtin.user:
    name: ansible
    groups: sudo
    append: true
```

Idempotent: if already a member, no-op. The role owns this prerequisite rather than depending on bootstrap variations.

---

v2.4

Title: Duo MFA on sudo for AD sudo group

Commit: `7eaf35a`

Changes:

- New template: `pam_sudo.j2` — PAM stack for `/etc/pam.d/sudo` with pam_duo, `pam_succeed_if user notingroup sg_sudo` carve-out, common-auth/account/session-noninteractive includes.
- New sudoers drop-in: `/etc/sudoers.d/sudo_timestamp_timeout` setting `Defaults timestamp_timeout=30`.
- New tag: `sudo-mfa` on the sudo PAM tasks.
- New defaults: `duo_sudo_enabled: true`, `sudo_timestamp_timeout: 30`.

- Validation: `grep -c "pam_duo.so" /etc/pam.d/sudo` and a runtime `sudo -n true` as the ansible user.

Compliance reference: Essential Eight ML2 — MFA for privileged users performing privileged actions. The only compliance reference in the role source code.

---

v2.3

Title: Duo MFA via duo-unix from Duo's official repo

Commit: `9d11756` (initial: `e02e4ac`)

Changes:

- New task file: `duo.yml` — GPG key fetch, APT repo add, legacy `libpam-duo`/`libduo3` purge, `duo-unix` install.
- New templates: `pam_duo.conf.j2` (with vault credentials), `pam_sshd.j2` (PAM stack for sshd).
- SSH `AuthenticationMethods` default changed to `publickey,keyboard-interactive`.
- New defaults: `duo_failmode: safe`, `duo_pushinfo: yes`, `duo_prompts: 3`, `duo_autopush: yes`, `break_glass_user: pbr_admin`.
- Why not Ubuntu universe `libpam-duo`: outdated 1.11.3 (2022) version, incompatible with current Duo Auth API, doesn't support April 2026 CA bundle rotation.

---

v2.2.1

Title: Remove invalid `core_dumpable` from sssd.conf.j2

Commit: `016259c`

Changes: Removed the `core_dumpable = false` directive from the SSSD config template — not a valid sssd.conf option, was silently being ignored.

---

v2.2

Title: krb5 udp_preference_limit, explicit ldap_id_mapping

Commits: `43a1aa5`, `4032534`

Changes (canary learnings from pbr-uisp-kl1):

- krb5.conf: added `udp_preference_limit = 0` to force TCP for Kerberos — addresses UDP packet size issues with large PAC (users in many groups).
- sssd.conf: explicit `ldap_id_mapping = True` — was implicit, made explicit for reviewability.
- General SSSD/PAM/sshd alignment tweaks discovered during canary deployment.

---

v2.1

Title: Drop ssh_local_access group; sudo group is the local gate

Commit: `0bdccfa`

Changes: Earlier versions referenced a custom `ssh_local_access` group for the local-account allow path. Simplified to use the standard local `sudo` group instead — one fewer thing to manage during bootstrap.

---

v2.0

Title: Baseline pre-canary-deploy

Commit: `f681246`

Description: The first version considered complete enough for canary deployment. v1 series was scaffolding (`96c3f79` initial structure, `11e8ee9` inventory, `44bf79e` vault + group_vars).

Deferred Items (Planned for v2.5)

These items have been identified during the v2.4 → v2.4.2 development cycle but deferred to keep the immediate release focused:

Item	Rationale to defer
CIS audit rules baseline (auditd rule file deployment)	Need to scope which CIS Linux Workstation/Server Profile applies. Useful but not blocking baseline operation.
Audit log forwarding to Graylog (auditd → audisp-remote)	Closes the LXC compliance gap if combined with auditd-on-KVM. Requires Graylog input config and a forwarder package decision.
`verify.yml` `vars_files` import for defaults inheritance	Currently verify.yml duplicates the `manage_auditd` logic from packages.yml. Cleaner via shared defaults file, but works correctly as-is.

Item	Rationale to defer
`retries: 2, delay: 30` on the realm join task	Would mask the multi-master replication lag pattern from operator view. Tension between operator visibility and automation smoothness.
Refactor `manage_auditd: 'auto'` sentinel	The string sentinel mixed into a boolean variable is awkward. Could be split into `manage_auditd: true|false` with a separate `manage_auditd_auto_skip_lxc: true` guard. Cosmetic; current logic is correct.

Where to Read Next

- **Overview & Repository Layout** — if you've reached this page first, start here
- **Deployment Runbook — New Host** — the standard procedure
- **Architecture & Design Decisions** — the "why" behind everything in the role

Revision #2
Created 2026-05-13 05:35:29 UTC by PBR_Documentation
Updated 2026-06-17 07:58:48 UTC by PBR_Documentation