

FIELD NOTES FROM A PRESALES SOLUTIONS ARCHITECT

The Root Password Changed at 10:17 and Nobody Logged In — The VM Console Path Your Firewall Never Sees

A presales war story about a Linux VM whose root password changed with no login in any log — the swapped guest-tools agent behind it, the password stealer that came with it, and the checks I now run before anyone types a password on a suspect server.

Daanial Farrukh · 27 October 2026 · <https://quabin.com/root-password-changed-no-login-vm-console-backdoor/>

The message came late one morning.

"The root password doesn't work anymore. Nobody changed it."

The server was a customer's Ubuntu 24.04 VM on a hosted private cloud. A web server with a handful of sites, SSH on a non-standard port, a firewall in front of it that had been reviewed twice that year.

Their first theory came fast.

"Someone brute-forced SSH. Check the firewall logs."

I asked one question.

"When was the last successful SSH login as root?"

They looked. "Two days ago. Ours."

"And the password changed this morning?"

"10:17."

A pause. "So how did they get in?"

That is the question this whole article is about. Because nobody got in the way anyone in that room expected. Nobody got in through the network at all.

The Trap, Named

That's the trap. When root changes under you, everyone stares at the network door — SSH, the firewall, the web application firewall, the panel.

A VM has a second door that is not on the network at all: the hypervisor console. It hides in plain sight because it is the door the cloud provider uses. Nobody treats it as an attack path, so nobody watches it.

What Actually Happened — In Order

It took two nights of log work to put the chain together. Here it is in one line:

Cloud portal login → VM console opened → old build-time user logs in → sudo → three backdoors installed → guest agent replaced → "reset password" called from inside the VM → root changed at 10:17

Now each link, with where the evidence was.

The portal login was not the customer's. The cloud portal's audit showed a login to the customer's tenant from an address that was not the customer's office or their admin's home. Minutes later the web console to the VM was opened.

Here is the part that trips people up. A console session leaves no trace in `auth.log` by itself. But it does leave one in the kernel log. When the hypervisor attaches a console, the guest sees a virtual USB tablet — the console's mouse — plug in. That one kernel line dated the entry to the second.

Root failed on the console. A forgotten user did not. On `tty1`, two failed root logins. Then a login as a user created when the VM was first built, months earlier, whose password had never been changed. It worked. `sudo su`. Root, in under three minutes, with a password that was probably in some old build document.

Three persistence mechanisms in twenty minutes. Once root, the attacker did not do one thing. They did three.

- A fake service: a static binary at `/usr/sbin/kernel-agent` with a systemd unit called `kernel-agent.service`, running as root. No package owned it.
- A password stealer in `/etc/ld.so.preload`: a shared object with a glibc-sounding name, loaded into every process on the system, hooking the functions that read passwords from a terminal.
- A PAM backdoor: one line added to `common-auth`, `sshd`, `su` and `sudo`, plus a `tty` variant in `login`:

```
auth optional pam_exec.so quiet expose_authtok /usr/libexec/glibc-langpack
```

`expose_authtok` hands the plaintext password to that script on every authentication. The attacker left a marker comment in each file and saved the original files in a hidden folder under `/var/lib/private/` — their own backup, so they could restore the files if they ever needed to look clean.

The guest-tools agent was swapped. Every cloud platform ships a small agent inside the VM that the portal talks to over localhost: reset the password, run a script, report the IP. For about an hour the attacker probed that agent from inside the VM — the signature checks on its API kept failing. Then they replaced the binary with a modified copy and left a `.bak` next to it. The running file's hash no longer matched the copy the vendor's package ships.

Then they cleaned up. `/root/.bash_history` emptied. Old records cut out of `/var/log/wtmp`, so last started two days late. The relevant lines deleted from `auth.log`. They missed one place: `journald` still had every line.

Hours later, 10:17. With no login of any kind — no SSH, no console, no panel — the journal shows calls to the agent's `reset-password` and `run-script` endpoints from localhost, and at 10:17:12 `chpasswd` changed root. The attacker did not need to be logged in anymore. The swapped agent did it for them, on a schedule.

Why the Usual Checks Missed It

The firewall was never in the path. The console goes hypervisor → VM, not network → VM. The distributed firewall, the port knocking, the custom SSH port — all irrelevant to that door.

The logs people trust had been edited. `auth.log` was missing lines. `wtmp` had been truncated. `last` and `who` showed a quiet server, not because the server was quiet but because the file that would have said otherwise had been cut.

The file timestamps lied. The attacker set the modification time of every file they touched back to an old date. `ls -l` showed nothing recent. `find -mtime` found nothing. That is why the PAM backdoor was found a full day after the first cleanup — it was caught by a search on change time, which a normal user cannot fake without changing the system clock.

And the first-page checks were all green. No extra UID 0 user. Root's crontab untouched. No new SSH keys. Kernel not tainted. Everything a standard "am I hacked" checklist starts with was clean.

Absence of evidence in a log the attacker could edit is not evidence of absence.

How to Actually Find This

Everything below is read-only. Run it from your own terminal, and — this matters — do not type a password on the suspect server until Step 3 comes back clean. If the stealer is loaded, every password typed is captured.

Step 1: Ask the journal, not the file.

```
journalctl --since "-3 days" _COMM=chpasswd
```

On this server it printed password changed for root with a time nobody could explain. Then compare what the journal knows about logins with what the file knows:

```
journalctl --since "-3 days" | grep -c "session opened for user"
```

```
grep -c "session opened for user" /var/log/auth.log
```

If the journal has more lines than the file for the same period, the file was edited. On this server it had.

Step 2: Find console sessions in the kernel log.

```
journalctl -k --since "-3 days" | grep -i "usb"
```

A virtual tablet or mouse attaching, with no physical change to the VM, is a console being opened from the hypervisor. This is how we knew the exact minute the attacker arrived — and that they arrived through the portal, not the network.

Step 3: Check the two places that capture passwords.

```
cat /etc/ld.so.preload
```

The file should not exist, or should be empty. Any library listed there is loaded into every process on the box. On this server it named one `.so` in `/usr/libexec/` that no package owned.

```
grep -r "pam_exec" /etc/pam.d/
```

A pam_exec.so line with expose_authtok that you did not add is a password stealer. Full stop.

This error is not a bug. It is the verification. If both of those come back clean, passwords typed on the server are safe. If either is dirty, nothing typed on this server from the moment of compromise can be trusted.

Step 4: Search by change time, not modification time.

```
find / -xdev -type f -ctime -3 -printf '%CY-%Cm-%Cd %CH:%CM %p\n' | sort
```

Modification time is one touch away from any date. Change time is set by the kernel when the inode changes and cannot be set from userspace. Sorted by ctime, the attacker's whole session appeared in order: the PAM files, the .so, the agent, the hidden backup folder — all within one twenty-minute window, no matter what ls -l said.

Step 5: Find services that no package owns.

```
systemctl list-units --type=service --state=running --no-pager
```

For any name you do not recognise:

```
dpkg -S /usr/sbin/kernel-agent
```

dpkg-query: no path found matching pattern means no package put that file there. A root-running service with a generic name and an unowned binary is yours to explain.

Step 6: Hash the guest agent against its own package copy.

Vendor tools usually keep a pristine copy of the agent in an architecture folder next to the installed one. Hash both:

```
sha256sum /usr/local/<vendor-tools>/agent /usr/local/<vendor-tools>/x64/agent
```

They must match. On this server they did not, and an agent.bak sat beside them with the original hash. Also check the vendor's own update log: no update had been applied, so the only way the running hash could differ was a manual swap.

Step 7: Find the user who should not exist.

```
awk -F: '$3>=1000 && $7!~/nologin|false/' /etc/passwd
```

Then for each one:

```
chage -l <user>
```

"Last password change" from the week the VM was built is the door. That user also sat in sudo, adm and lxd — three groups, any one of which is root.

How to Actually Clean It — In This Order

The order matters because of the stealer. **Clear the password capture first, then rotate passwords, never the other way round.** Rotating first hands the attacker the new password.

- 1 **Snapshot the VM from the portal.** That is your evidence. Everything after this changes state.

- 2 **Copy evidence out.** The unowned binaries, /etc/pam.d/, the .so, the agent and its .bak, the journal export. Tar it, pull it off the server over your own key, hash the tarball.
- 3 **(changes state) Remove the preload.** rm /etc/ld.so.preload, move the .so to a quarantine folder. Nothing typed after this is captured by that component.
- 4 **(changes state) Restore PAM.** Diff every file in /etc/pam.d/ against the attacker's own backup copies or against the package versions, and put the originals back. The attacker's backup folder made this fast; dpkg -V libpam-runtime confirms it.
- 5 **(changes state) Remove the fake service.** systemctl disable --now kernel-agent, then remove the unit and the binary to quarantine.
- 6 **(changes state) Restore the guest agent.** Reinstall the vendor tools from the package, confirm the hash matches the package copy, restart the agent.
- 7 **(changes state) Lock the entry user.** usermod -L -e 1 -s /usr/sbin/nologin <user> and deluser <user> sudo — and the same for adm and lxd.
- 8 **Now rotate every password.** Root, every sudo user, the panel, the database. On this server the portal's own "reset password" failed, so root was set from the installer ISO: boot the Ubuntu installer, Help > Enter shell, mount the LVM root at /mnt, chroot /mnt passwd root.
- 9 **Re-run Steps 1 to 7.** On this server a full second pass — unowned binaries, UID 0, cron, timers, kernel modules, webshells, authorized keys — came back clean.
- 10 **Plan the rebuild.** A cleaned server is a server you hope is clean. A rebuilt server is one you know is clean. Treat the cleanup as the bridge to a rebuild, not as the end.

Two things I can say with certainty from the evidence, and one I cannot.

The stealer stored what it captured in a hidden file under /var/log, encrypted with a key kept beside it — it did not send anything out over the network by itself. It had two records, both from passwords that were changed anyway.

The attacker's session proved outbound internet worked: a Python package install ran from the server during their window, with no login at that time. **A compromised server with open egress is a staging post. Egress filtering is not a nice-to-have on a web server.**

What I cannot tell you is how the portal credential got into someone else's hands. The provider's own platform logs and the firewall records for that address were still being pulled when I wrote this. I will not guess at it here, because a guess presented as a cause is how the next incident gets missed.

The Second Server

Within a day, a second VM at another customer on the same platform was reported with the same symptoms. Different OS family, same pattern: portal console, build-time user, the same three backdoors.

We wrote the checks above into a read-only script and ran that first, before anyone touched the server. Then the snapshot, then the evidence, then the quarantine, then the agent, then the lock, and only then the passwords.

Plain takeaway: when the first server tells you the attacker's method, the second server is a checklist, not an investigation.

The Bigger Lesson — Six Checks for Any Hosted VM

- 1 **The console is a login path. Treat portal accounts like root.** MFA on the cloud portal, named users only, and an alert on every console open. `journalctl -k | grep -i tablet` tells you when one happened.
- 2 **No build-time users survive go-live.** `awk -F: '$3>=1000' /etc/passwd` and `chage -l` for each. Expire or remove what nobody owns.
- 3 **/etc/ld.so.preload must not exist.** `test -s /etc/ld.so.preload && echo ALERT` in a daily job that mails you.
- 4 **PAM under change control.** `sha256sum /etc/pam.d/* > /root/pam.baseline` on a clean day, then `sha256sum -c /root/pam.baseline` on a schedule.
- 5 **Hash the guest agent.** `sha256sum` of the running binary against the package copy, on the same schedule.
- 6 **Filter egress.** A web server needs to reach its update mirror and its DNS resolvers. It does not need to reach the whole internet. Prove it with `curl -m 5 https://pypi.org` from the server — it should fail.

Each one is a minute. Together they close the door this attacker walked through, and the two they would have used next.

What I Tell Every Customer Now

When a customer shows me a VM on any hosted platform, I ask one question:

"Who can open a console to this VM from the portal, and would you know within the hour if they did?"

If the answer is "these three people, with MFA, and the console event alerts us", they're in good shape. If the answer is "the provider handles the portal", 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

Why did the root password change when nobody logged in?

Because the change was made by a program, not a person. A tampered guest-tools agent inside the VM accepted a "reset password" call from localhost and ran `chpasswd`. Look for `journalctl _COMM=chpasswd` to see the exact time, and then check whether the agent binary matches the vendor package.

How can someone access a VM without going through the firewall?

Through the hypervisor console in the cloud portal. The console is attached by the hypervisor directly to the VM, like a keyboard and screen plugged into a physical server. It never crosses the VM's network interface, so no firewall, security group or SSH setting applies to it.

What is `ld.so.preload` and why is it dangerous?

It is a file that lists shared libraries the dynamic loader must inject into every program on the system. Legitimate systems almost never use it. An attacker can list a library there that hooks password-reading functions and captures every password typed, in every program, without any process looking unusual.

What does pam_exec expose_authtok do?

The pam_exec.so module runs an external program during authentication. With expose_authtok, PAM passes the user's plaintext password to that program on standard input. Added to common-auth, sshd, su and sudo, it captures every password used on the server. Any such line you did not add yourself is a backdoor.

Why search files by ctime instead of mtime?

Modification time can be set to any value with touch -d, and attackers do this to hide. Change time is updated by the kernel whenever the inode changes and cannot be set from userspace without changing the system clock. find / -xdev -ctime -3 shows what really changed, in order.

Should I clean a compromised server or rebuild it?

Clean it first so you can collect evidence, stop password capture and get control back, then rebuild. A cleanup proves what you found. A rebuild from a known-good image is the only thing that proves nothing else is there.

Read online: <https://quabin.com/root-password-changed-no-login-vm-console-backdoor/>