SECURITY INCIDENT RESPONSE · 29 AUGUST 2026

Ransomware compromise of a virtualization host

RANSOMWARE | HYPERVISOR | ROOTKIT | BACKUP INTEGRITY | FORENSICS | FULL RECOVERY

01 Executive summary

On 29 August 2026, the UPNIX team detected and contained a ransomware attack against a client's virtualization host. An intruder obtained administrative access through the hypervisor's web management interface, encrypted guest disk images and the on-host backups, and deployed a cryptominer, a userland rootkit and an SSH backdoor while systematically erasing logs. Through disciplined forensics and one untouched offsite backup copy, the UPNIX team reconstructed the full timeline, removed all persistence, prevented a scheduled job from destroying the only clean backups, and rebuilt the host from a trusted baseline. No ransom was paid; guests were restored from verified offsite copies.

02 Environment

A single virtualization host (Proxmox VE) running roughly nine guest virtual machines — databases, application and editor nodes, a VPN endpoint, plus staging and development systems. The UPNIX team provided agent-based monitoring and operations. Backups were written both to the host itself and to an offsite, S3-compatible object-storage bucket.

03 Detection and first response

The UPNIX team's monitoring raised a cascade of alerts within minutes: several guests unreachable, sustained high CPU on the host, and a block of missing-data signals. The on-call responder connected and found an encryption process *actively running*, disguised as a system-information utility, launched interactively under the root account, consuming ~165 % CPU with its on-disk binary already deleted. The host was network-isolated immediately to stop further damage and lateral movement.

04 Timeline (host local time, UTC)

09:18	Administrative login to the hypervisor web interface under the root realm account. The log shows a password “change” — in fact a side effect of the abused flaw; the original password kept working throughout.
09:34	Root shell opened through the web console. The intruder patched the hypervisor's authentication module, deleted the process viewers (`htop` , `top`), shut down every guest and set them to not auto-start.
09:36	Rotated logs in the system log directory zeroed; `cron` and all `systemd` timers disabled.
09:39	On-host VM backups encrypted *first* — before the live images — to deny an easy local restore.
~09:40	Live guest disk images encrypted (a custom extension appended) and ransom marker files written to the root directory.
10:07–10:09	Second wave: a cryptominer installed as a disguised “node health” service; a userland `LD_PRELOAD` rootkit hiding processes, ports and paths; journald set to store nothing; login logs (`wtmp/btmp/lastlog/auth.log`) replaced with symlinks to `/dev/null` ; and an SSH backdoor added via an `AuthorizedKeysCommand` inside a `Match User root` block.

05 Entry vector

The intrusion was **not** a stolen SSH key and **not** a brute-forced password: the original administrative password continued to work, and external brute-force attempts only ever received authentication failures. Evidence pointed to a weakness in the hypervisor's web/API management layer, which was running an **end-of-life release**. The decisive indicator came from the attacker's own behaviour: sixteen minutes after entry they modified the authentication module to *harden* the very check they had just abused — closing the door behind themselves so no one else could follow. Defenders should read this as a reminder that the management/API plane of a virtualization platform is a crown-jewel asset and must never run past its supported life.

06 Anti-forensics observed

- File timestamps were forged with `touch -r`; the copied times matched to the nanosecond across unrelated files — an anomaly that is itself evidence of tampering.
- The true chronology was recovered from inode change time (`ctime`), which cannot be forged without moving the system clock.
- Decoy and implant files were marked immutable and append-only (`chattr +i/+a`), so a normal delete returned "Operation not permitted" and could be misread as a filesystem fault.
- The platform's clustered authentication log, held in an in-memory cluster filesystem, survived the wipe of on-disk logs and preserved the genuine login records used to anchor the timeline.

> **Near-miss: the backup that would have destroyed itself**
>
> The offsite copy — written about sixteen hours before the attack — was intact. But a one-way mirror job (`mirror --remove`) was scheduled to run roughly fifty hours later. Because the source now held only encrypted files, that job would have deleted every clean backup and replaced it with encrypted versions, and the bucket was un-versioned, so the loss would have been irreversible. the UPNIX team disarmed the job, enabled versioning, and scoped the storage credential to deny object deletion.

07 Containment and recovery

Because the userland rootkit injected into every new shell session, ordinary tooling on the host could no longer be trusted. Rather than attempt an in-place clean-up, the UPNIX team treated the host as a total loss and **rebuilt it from a trusted baseline**. All five persistence units (miner service, watchdog timer, maintenance and update units) were removed; a clean environment started with `env -i LD_PRELOAD=` confirmed zero residual processes, timers or preloads. One live database image survived intact — it happened to be held open at the moment the attacker's process was killed, so its encryption never completed — and the remaining guests were restored from the verified offsite copies.

08 Lessons learned

- **Retire end-of-life management planes.** The web/API layer of a hypervisor is the highest-value entry point; keep it current and access-restricted.
- **Freeze outbound copy jobs after any encryption event.** A one-way sync from a poisoned source is a self-inflicted second strike. Inventory every job that writes to external copies and stop it before investigating.
- **Make backups tamper-resistant.** Enable versioning and least-privilege (no delete) on backup storage; a credential shared across hosts turns one compromise into many.
- **Treat a userland rootkit as loss of tool integrity.** Re-verify findings from a clean environment and prefer rebuild over clean-up.

- **Use package verification for triage.** On Debian/Ubuntu, `dpkg -V` lists every file that no longer matches its package — the fastest way to spot a replaced system binary or library, even when timestamps are forged.
- **Build timelines on `ctime`, not `mtime`.** Nanosecond-identical modification times across unrelated files indicate tampering.
- **Check the *effective* SSH configuration.** A backdoor via `AuthorizedKeysCommand` is invisible to a plain `sshd -T`; inspect it with `sshd -T -C user=root` and review `Match` blocks.
- **Prefer push-based monitoring.** An agent that reports outward keeps delivering metrics even when inbound access to a host is cut during isolation.

This report has been fully anonymized. All client identifiers — organization names, hostnames, domains, IP addresses, account names and storage locations — have been removed. Technical findings, methods and recommendations are retained unchanged so the case remains useful to other operators.

About UPNIX Technologies. The UPNIX team builds, monitors, protects and keeps running the infrastructure businesses depend on — security, 24/7 monitoring and operations for clients worldwide, alongside software and AI engineering.
