What is cron job abuse?

Cron job abuse is privilege escalation through scheduled tasks that run as root but trust attacker-controlled input. If a root cron job executes a world-writable script, uses a wildcard an attacker can poison, or relies on a relative path you can hijack, a low-privileged user can make root run their code on the next schedule. Inspecting `/etc/crontab`, the cron directories, and the permissions of every scripted job is the way to find it.

HOW IT WORKS

01 The common setups and payload

The attacker inspects scheduled jobs and looks for one they can influence:

- Read schedules: `cat /etc/crontab` and list `/etc/cron.*` and `/var/spool/cron`
- World-writable script run by root: append a reverse shell or `chmod +s /bin/bash`, then wait for the run.
- Wildcard injection: a root cron running `tar czf backup.tar.gz *` in a writable directory can be hijacked by creating files named like `--checkpoint=1` and `--checkpoint-action=exec=sh runme.sh`.
- Relative path: a cron calling a binary by name is abusable via PATH hijacking.

Documented techniques shown for defenders.

> CHECK WRITABILITY
>
> *For every root cron job, check the permissions of the script it runs and the directory it runs in. A world-writable script or working directory is a direct path to root on the next tick.*

HOW TO DEFEND

- Lock down permissions on every script a root cron job runs (root-owned, not world-writable) and on its working directory.
- Avoid wildcards in privileged cron commands, or use full paths and `--` to end option parsing.
- Use absolute paths for binaries in cron to prevent PATH hijacking.
- Review all cron jobs and remove unnecessary ones.
- Monitor cron scripts and directories for unexpected changes.

SOURCES

[1] Linux man-pages: crontab(5)

[2] MITRE ATT&CK: Privilege Escalation (TA0004)

[3] NIST SP 800-115 Technical Guide to Security Testing

Find the privilege-escalation paths before an attacker does.

securelayer7.net/learn/privilege-escalation/what-is-cron-job-abuse

Open online