Linux Privilege Escalation via Cron Job Misconfigurations

Research Summary & Attack Scenarios

Introduction

This research explores how misconfigured cron jobs on Linux systems can allow local privilege escalation (LPE) — enabling an attacker with low privileges to gain full root access. While not a CVE or 0day, these findings are extremely relevant for both enterprise and cloud environments, where default or legacy configurations are still widespread.

Methodology

- Over 200 hours of lab-based research
- Manual review and automated audit of real-world cron configurations across Ubuntu, OpenSUSE, Oracle Linux, and other distributions
- Development of custom audit/detection scripts
- Hands-on exploitation and root access scenarios

Attack Scenarios

1. World-Writable Script Paths

Pattern:
Cron executes scripts in directories with permissions such as 777.

Exploitation:
A regular user replaces the target script, causing it to run as root on the next cron execution.

2. Relative Path Execution

Pattern:
Cron job calls a script without specifying the full path.

Exploitation:
Attacker places a malicious script with the same name in a PATH directory. Cron executes the attacker's code as root.

3. Insecure Temporary Files

Pattern:
Scripts create predictable or insecure temp files in /tmp.

Exploitation:
Attacker can symlink the temp file, escalate privileges, or read sensitive data via race condition.

4. Environment Variable Injection

Pattern:
Cron jobs rely on user-controllable variables in the command line.

Exploitation:
Attacker redefines the environment variable, causing cron to execute arbitrary code as root.

Impact

- Full root access from unprivileged user
- Persistent root-level backdoors (example: custom SUID-root binary “rootbash”)
- Potential for ransomware, lateral movement, and data exfiltration

Tools & Scripts

- audit-cron.sh: Automated audit script to detect risky cron jobs and permissions
- secure-cron-template.sh: Baseline script template for secure cron usage
- monitor-cron-changes.sh: File integrity monitoring for cron jobs and scripts

Real-World Example: The “rootbash” Binary

During research, we identified a SUID-root shell (rootbash) left behind after exploitation. This binary provided root shell access on all tested platforms and demonstrated the persistent risk associated with cron job misconfigurations.

For detailed static/dynamic analysis, see docs/ROOTBASH_ANALYSIS.md.

Detection and Mitigation

- **Immediate audit:** Use the tools provided in /tools
- **Hardening:** Apply all recommendations from docs/HARDENING.md
- **Detection:** See docs/DETECTION.md for SIEM rules and monitoring scripts

Responsible Disclosure

All findings were shared with relevant vendor security teams. No production systems were targeted or harmed.

Author: Nelson Adhepeau
Contact: privexploits@protonmail.com
LinkedIn: linkedin.com/in/nelson-adhepeau

For educational and defensive security use only.

This PDF is included as a technical executive summary for blue teams, pentesters, and IT leads. For the full detailed research, see the main README on GitHub.