**Linux Hardening**

Proper Linux hardening can eliminate most, if not all, opportunities for local privilege escalation. The following steps should be taken, at a minimum, to reduce the risk of an attack being able to elevate to root-level access:

**Updates and Patching**

Many quick and easy privilege escalation exploits exist for out-of-date Linux kernels and known vulnerable versions of built-in and third-party services. Performing periodic updates will remove some of the most "low hanging fruit" that can be leveraged to escalate privileges. On Ubuntu, the package [unattended-upgrades](https://packages.ubuntu.com/jammy/admin/unattended-upgrades) is installed by default from 18.04 onwards and can be manually installed on Ubuntu dating back to at least 10.04 (Lucid). Debian based operating systems going back to before Jessie also have this package available. On Red Hat based systems, the [yum-cron](https://man7.org/linux/man-pages/man8/yum-cron.8.html) package performs a similar task.

**Configuration Management**

This is by no means an exhaustive list, but some simple hardening measures are to:

* Audit writable files and directories and any binaries set with the SUID bit.
* Ensure that any cron jobs and sudo privileges specify any binaries using the absolute path.
* Do not store credentials in cleartext in world-readable files.
* Clean up home directories and bash history.
* Ensure that low-privileged users cannot modify any custom libraries called by programs.
* Remove any unnecessary packages and services that potentially increase the attack surface.
* Consider implementing [SELinux](https://www.redhat.com/en/topics/linux/what-is-selinux), which provides additional access controls on the system.

**User Management**

We should limit the number of user accounts and admin accounts on each system, ensure that logon attempts (valid/invalid) are logged and monitored. It is also a good idea to enforce a strong password policy, rotate passwords periodically, and restrict users from reusing old passwords by using the /etc/security/opasswd file with the PAM module. We should check that users are not placed into groups that give them excessive rights not needed for their day-to-day tasks and limit sudo rights based on the principle of least privilege.

Templates exist for configuration management automation tools such as [Puppet](https://puppet.com/use-cases/configuration-management/), [SaltStack](https://github.com/saltstack/salt), [Zabbix](https://en.wikipedia.org/wiki/Zabbix) and [Nagios](https://en.wikipedia.org/wiki/Nagios) to automate such checks and can be used to push messages to a Slack channel or email box as well as via other methods. Remote actions (Zabbix) and Remediation Actions (Nagios) can be used to find and auto correct these issues over a fleet of nodes. Tools such as Zabbix also feature functions such as checksum verification, which can be used for both version control and to confirm sensitive binaries have not been tampered with. For example, via the [vfs.file.cksum](https://www.zabbix.com/documentation/4.0/manual/config/items/itemtypes/zabbix_agent) file.

**Audit**

Perform periodic security and configuration checks of all systems. There are several security baselines such as the DISA [Security Technical Implementation Guides (STIGs)](https://public.cyber.mil/stigs/) that can be followed to set a standard for security across all operating system types and devices. Many compliance frameworks exist, such as [ISO27001](https://www.iso.org/isoiec-27001-information-security.html), [PCI-DSS](https://www.pcisecuritystandards.org/pci_security/), and [HIPAA](https://www.hhs.gov/hipaa/for-professionals/security/index.html) which can be used by an organization to help establish security baselines. These should all be used as reference guides and not the basis for a security program. A strong security program should have controls tailored to the organization's needs, operating environment, and the types of data that they store and process (i.e., personal health information, financial data, trade secrets, or publicly available information).

An audit and configuration review is not a replacement for a penetration test or other types of technical, hands-on assessments and is often seen as a "box-checking" exercise in which an organization is "passed" on a controls audit for performing the bare minimum. These reviews can help supplement regular vulnerability scanning and penetration testing and strong patch, vulnerability, and configuration management programs.

One useful tool for auditing Unix-based systems (Linux, macOS, BDS, etc.) is [Lynis](https://github.com/CISOfy/lynis). This tool audits the current configuration of a system and provides additional hardening tips, taking into consideration various standards. It can be used by internal teams such as system administrators as well as third-parties (auditors and penetration testers) to obtain a "baseline" of the system's current security configuration. Again, this tool or others like it should not replace the manual techniques discussed in this module but can be a strong supplement to cover areas that may be overlooked.

After cloning the entire repo, we can run the tool by typing ./lynis audit system and receive a full report.

htb\_student@NIX02:~$ ./lynis audit system

[ Lynis 3.0.1 ]

################################################################################

Lynis comes with ABSOLUTELY NO WARRANTY. This is free software, and you are

welcome to redistribute it under the terms of the GNU General Public License.

See the LICENSE file for details about using this software.

2007-2020, CISOfy - https://cisofy.com/lynis/

Enterprise support available (compliance, plugins, interface and tools)

################################################################################

[+] Initializing program

------------------------------------

###################################################################

# #

# NON-PRIVILEGED SCAN MODE #

# #

###################################################################

NOTES:

--------------

\* Some tests will be skipped (as they require root permissions)

\* Some tests might fail silently or give different results

- Detecting OS... [ DONE ]

- Checking profiles... [ DONE ]

---------------------------------------------------

Program version: 3.0.1

Operating system: Linux

Operating system name: Ubuntu

Operating system version: 16.04

Kernel version: 4.4.0

Hardware platform: x86\_64

Hostname: NIX02

The resulting scan will be broken down into warnings:

Warnings (2):

----------------------------

! Found one or more cronjob files with incorrect file permissions (see log for details) [SCHD-7704]

https://cisofy.com/lynis/controls/SCHD-7704/

! systemd-timesyncd never successfully synchronized time [TIME-3185]

https://cisofy.com/lynis/controls/TIME-3185/

Suggestions:

Suggestions (53):

----------------------------

\* Set a password on GRUB boot loader to prevent altering boot configuration (e.g. boot in single user mode without password) [BOOT-5122]

https://cisofy.com/lynis/controls/BOOT-5122/

\* If not required, consider explicit disabling of core dump in /etc/security/limits.conf file [KRNL-5820]

https://cisofy.com/lynis/controls/KRNL-5820/

\* Run pwck manually and correct any errors in the password file [AUTH-9228]

https://cisofy.com/lynis/controls/AUTH-9228/

\* Configure minimum encryption algorithm rounds in /etc/login.defs [AUTH-9230]

https://cisofy.com/lynis/controls/AUTH-9230/

and an overal scan details section:

Lynis security scan details:

Hardening index : 60 [############ ]

Tests performed : 256

Plugins enabled : 2

Components:

- Firewall [X]

- Malware scanner [X]

Scan mode:

Normal [ ] Forensics [ ] Integration [ ] Pentest [V] (running non-privileged)

Lynis modules:

- Compliance status [?]

- Security audit [V]

- Vulnerability scan [V]

Files:

- Test and debug information : /home/mrb3n/lynis.log

- Report data : /home/mrb3n/lynis-report.dat

The tool is useful for informing privilege escalation paths and performing a quick configuration check and will perform even more checks if run as the root user.

**Conclusion**

As we have seen, there are various ways to escalate privileges on Linux/Unix systems - from simple misconfigurations and public exploits for known vulnerable services to exploit development based on custom libraries. Once root access is obtained, it becomes easier to use it as a pivot point for further network exploitation. Linux (and all system) hardening is critical for organizations of all sizes. Best practice guidelines and controls exist in many different forms. Reviews should include a mix of hands-on manual testing and review and automated configuration scanning and validation of the results.