MANDIANT

WHITE PAPER

LINUX ENDPOINT HARDENING TO PROTECT AGAINST MALWARE AND DESTRUCTIVE ATTACKS

Contents

Background

The Linux operating system and supporting applications are becoming a prime target for adversaries. Linux is used as the operating system backend for many components that automate facets of critical infrastructure, on-premises and cloud-based technologies, and Internet of Things (IoT) devices.

This document provides recommendations to protect Linux endpoints against adversary techniques such as lateral movement, privilege escalation, and deploying rootkits or modified kernel modules that possess either a malicious or destructive capability. Drovorub (https://media.defense.gov/2020/Aug/13/2002476465/-1/-1/0/CSA_DROVORUB_RUSSIAN_GRU_MALWARE_AUG_2020.PDF) is an example of a Linux malware toolset that includes a kernel module rootkit that can be leveraged for command and control (C2) communications, file download and upload capabilities, and the execution of arbitrary commands.

Similar to Windows-based architectures, security protections need to be aligned for Linux endpoints to harden credentials, access methods, protect the kernel, and bolster auditing and visibility of activities.

Platform Hardening

Kernel Module Signing

As of Linux kernel version 3.7, signed kernel modules using digital signatures can be leveraged to enhance the security of Linux by preventing unsigned and untrusted modules from loading and running. Kernel module signing can be configured within the .config file as part of the `CONFIG_MODULE_SIG` configuration parameters. This type of trust chaining for kernel modules works in parallel with UEFI Secure Boot being enabled, which protects the bootloader and firmware.

Example methods which can be used to configure trust-chains and enforce kernel module signing are referenced in Table 1.

TABLE 1. References to enable Secure Boot and Kernel Module Signing.

Linux Distributions	Reference Links
Linux Kernel	https://www.kernel.org/doc/html/v4.15/admin-guide/module-signing.html
Red Hat Enterprise Linux	https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/managing_monitoring_and_updating_the_kernel/signing-kernel-modules-for-secure-boot_managing-monitoring-and-updating-the-kernel
Ubuntu	https://ubuntu.com/blog/how-to-sign-things-for-secure-boot
Fedora	https://docs.fedoraproject.org/en-US/Fedora/23/html/System_Administrators_Guide/sect-signing-kernel-modules-for-secure-boot.html
Gentoo	https://wiki.gentoo.org/wiki/Signed_kernel_module_support
Oracle	https://www.oracle.com/technical-resources/articles/linux/signed-kernel-modules.html
Debian	https://wiki.debian.org/SecureBoot

`Sysctl` Configuration Parameters

`Sysctl.conf` is the main kernel configuration file on Linux systems. The `sysctl -p` command can be used to modify kernel parameters at runtime, which will s`tore the configuration parameters within the `/etc/sysctl.conf` file.

The `/etc/sysctl.conf` file can also be manually edited, with the added or removed parameters being loaded upon executing the `sysctl -p` command.

Example parameters to include within the `/etc/sysctl.conf` file to harden a Linux endpoint are referenced in Figure 1.

```
# Disables IP forwarding
net.ipv4.ip_forward = 0

# Disables packet redirect sending
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0

# Does not accept source routing
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0

# Enables reverse path filtering
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1

# Disables IPv6 router advertisements
net.ipv6.conf.all.accept_ra = 0
net.ipv6.conf.default.accept_ra = 0

# Logs suspicious packets
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.default.log_martians = 1

# Controls the System Request debugging functionality of the kernel
kernel.sysrq = 0

# Controls whether core dumps will append the PID to the core filename (useful for debugging
multi-threaded applications)
kernel.core_uses_pid = 1

# Enables TCP SYN Cookies
net.ipv4.tcp_syncookies = 1

# Enables SYN-flood protections
net.ipv4.tcp_synack_retries = 5

# Enables ExecShield protection
kernel.exec-shield = 2

# Enable full address space randomization
kernel.randomize_va_space = 2
```

FIGURE 1. Recommended sysctl configuration parameters.

Kernel Module Loading Enforcement

Within `sysctl.conf`, the Linux kernel can be configured to not allow for new modules to be loaded.

```
# Prevent new modules from loading
kernel.modules_disabled = 1
```

FIGURE 2. Setting to disable new kernel modules from loading using `sysctl`.

Another option for setting this configuration is referenced within Figure 3.

```
echo 1 > /proc/sys/kernel/modules_disabled
```

FIGURE 3. Setting to disable kernel module loading using `/proc`.

To change this configuration back to the default setting, the `kernel.modules_disabled` setting can be configured back to "`0`" – and a reboot will need to occur.

While this configuration can certainly reduce the attack surface from an adversary loading malicious kernels (e.g., rootkits), this can also impact legitimate applications that would need to be loaded after the hardened configuration is enforced.

Note: Some modules are not loaded at boot time – and may be loaded as part of normalized system operations. To ensure that legitimate and authorized modules can still be loaded, modules can be defined either within the `/etc/modules-load.d` file or within a startup script (Figure 4).

```
#!/bin/sh/
sleep <value>
insmod <module1> <module2> <module3>
echo 1 > /proc/sys/kernel/modules_disabled
```

FIGURE 4. Example startup script to load trusted modules.

SSH Attack Surface Reductions

Secure Shell (SSH) is a commonly used method to access and administer Linux endpoints. Adversaries often use SSH to gain initial access, maintain persistence, and laterally move.

Organizations should define policies and procedures to harden SSH access methods for Linux endpoints. The following recommendations should be considered:

- Least privileged access should be enabled for all SSH-accessible accounts and groups, especially for automated processes and remote access.
- To limit SSH access to Linux endpoints for only authenticated non-root user accounts, set `PermitRootLogin` to "`No`" in the `/etc/ssh/sshd_config` file.
- Implement user or group level restrictions for SSH access to Linux endpoints through `AllowUser`, `AllowGroup`, `DenyUser`, `DenyGroup` settings defined within the `/etc/ssh/sshd_config` file.
- `AuthenticationMethods` configuration parameters in the `/etc/ssh/sshd_config` file can be used to specify authentication methods that can be utilized. Organizations should consider defining more than one authentication method in order to enforce a type of multi-factor authentication (MFA) for accessing Linux endpoints.
- Cryptographic public key-based authentication (PKI) should be configured along with a strong password to enable a form of MFA to harden authentication access to a Linux endpoint when SSH is required.

  For this configuration, all users would need to create public and private key pairs, and the keys would need to be added to the endpoint before key-based authentication is enforced. The example configuration (Figure 5) within the `/etc/ssh/sshd_config` file will require a user to perform both password-based and key-based authentication, enforcing a form of MFA for connecting to the endpoint.

```
AuthenticationMethods publickey,password
```

FIGURE 5. Require both public key and password-based authentication.

- Host-based firewalls should be enforced to limit origination addresses that can connect to the SSH service on an endpoint. This is strongly recommended for any Linux endpoints that have SSH services exposed to the Internet or untrusted locations. Iptables and nftables are Linux utilities that can be used to configure IP packet filtering and limit SSH access from a specific set of systems.

  Alternatively, TCP wrappers could be used to implement packet filtering and limit origination systems that can access SSH services on Linux endpoints. TCP wrappers work by configuring two files: `/etc/hosts.allow and /etc/hosts.deny`. It is recommended to add "`ALL`" hosts to the hosts.deny file and only the specific authorized hosts to the hosts.allow file.
- If there is not a requirement to use X11 applications, set X11Forwarding to "`No`" in the `/etc/ssh/sshd_config` file.
- To limit the number of authentication attempts, set the MaxAuthTries setting to "`4`" in the `/etc/ssh/sshd_config` file.
- Set ClientAliveInterval between 300-900 seconds (5-15 minutes) in the `/etc/ssh/sshd_config` file and set `ClientAliveCountMax` to "`1`".
- To limit the risk of backdoor capabilities, set `AllowTcpForwarding` to "`No`" in the `/etc/ssh/sshd_config` file.
- To enhance auditing and visibility, set `LogLevel` to `VERBOSE` in the in `/etc/ssh/sshd_config` file.
- To disable accounts with empty passwords from being able to login, set `PermitEmptyPasswords` to "`No`" in the `/etc/ssh/sshd_config` file.

Cron Jobs Review and Permission Restrictions

Cron jobs are used to set scheduled tasks on a Linux endpoint. Cron jobs are often used by adversaries to establish a form persistence or to elevate permissions by invoking processes or modules to run within the context of the system. Organizations should review and enforce visibility for cron jobs configured on Linux endpoints.

The command referenced in Figure 6 can be used to list all the cron jobs lon a Linux endpoint for a specific user:

```
crontab -u <username> -l
```

FIGURE 6. Listing Cronjobs for a user.

The commands referenced in Figure 7 can be used to list all the cron jobs configured on an endpoint:

```
cat /etc/crontab
ls -al  /etc/cron.d
ls -al /etc/cron.hourly/
ls -al /etc/cron.daily/
ls -al /etc/cron.weekly
ls -al /etc/cron.monthly/
```

FIGURE 7. List all the cron jobs listed on a system.

The specific parameters associated with a cron job should also be reviewed, including any binaries or modules that re configured for execution, as these files should only be writable by the root user. The full permissions for any executable files configured as part of a cron job can be listed by using the command referenced in Figure 8.

```
ls -al <path of the binary>
```

FIGURE 8. Full permission listing for a binary.

Additionally, cron job file permissions should be reviewed. Read access to cron files could provide adversaries with the ability to gain insight on jobs that run on the endpoint, and could provide a method to gain unauthorized privileged access or create binaries or modules that will "blend in" with expected activities. Write access to cron files could provide unprivileged users with the ability to elevate privileges.

The command referenced in Figure 9 determines if the `/etc/crontab` file has the correct permissions.

```
stat -c "%a %u %g" /etc/crontab | egrep -v
".00 0 0"
```

FIGURE 9. Command to verify permissions of `/etc/crontab`.

If the above command emits no output, then the system is configured as recommended.

The command referenced in Figure 10 determines if the `/etc/cron.d` directory has the correct permissions.

```
stat -c "%a %u %g" /etc/cron.d | egrep -v
".00 0 0"
```

FIGURE 10. Command to verify permissions of `/etc/cron.d`.

If the above command emits no output, then the system is configured as recommended.

The commands referenced in Figure 11 will restrict access for non-root accounts to the `/etc/crontab` file and the `/etc/cron.d` directory.

```
chown root:root /etc/crontab
chmod og-rwx /etc/crontab
chown root:root /etc/cron.d
chmod og-rwx /etc/cron.d
```

FIGURE 11. Command to restrict read/write access to non-root accounts for cron files and directory.

SUID Executables

SUID (set owner USER ID on execution) is a file permission that can be configured so that files will execute under the permissions of the file owner. Files with SUID permissions are often used by adversaries to escalate privileges and perform malicious activities on a system. Although there are legitimate reasons to have a binary with SUID, it is important to identify and review SUID binaries on a regular basis.

Organizations should review all files on Linux endpoints that have the SUID bit set. Checking SUID should be part of the regular review process including post update/installation activity.

The commands referenced in Figure 12 can be used to search for SUID executables on a Linux endpoint.

```
find / -perm -u=s -type f 2>/dev/null
df --local -P | awk '{if (NR!=1) print $6}
find <partition> -xdev -type f -perm -4000
```

FIGURE 12. Commands to search for SUID executables.

The list of identified files should be reviewed for any binary that may allow a user to escalate privileges to root. Any file editor, compiler, or interpreter that can be used to read or overwrite a file should also be reviewed. If not required, SUID permissions should be removed for identified files.

Mounted Partitions Permissions Options

Once an adversary gains a foothold on a compromised endpoint with non-privileged access, as part of reconnaissance, an adversary may look for partitions with weak or misconfigured permission settings with the goal of executing binaries from storage locations for potential privilege escalation or lateral movement.

Organizations should review the existing configured options on mount points and evaluate the following conditions:

- Temporary storage mount points (e.g., `/tmp`, `/var/tmp`, `dev/shm`, removable media), especially world-writable directories, should be restricted from running executable binaries and from creating setuid files to avoid non-root users executing privileged programs or introducing potentially malicious software on the endpoint. Blocking the execution of binary files and creation of `setuid` files can be done by using "`noexec`" and "`nosuid`" mount options.
- Temporary storage mount points are not intended to support block or character special device files, and users should be restricted from creating these filetypes. Executing character or block special device filetypes from file systems increase the opportunity for unprivileged users to elevate permissions. Blocking the creation of special device filetypes can be accomplished using the "`nodev`" mount option.
- Directories (e.g., `/var/log`, `/var/log/audit`) where log data is usually stored are recommended to have a separate partition configured to protect against resource exhaustion issues (where logs can significantly grow), is minimal, and the storage capacity is sized appropriately, creating a separate partition can be planned at a later stage or during a scheduled maintenance window.

TABLE 2. Recommended mount point options.

Mount Point	Mount Options
/tmp	nosuid, noexec, nodev
/home	nodev
/dev/shm	nosuid, noexec, nodev
/var	defaults
/var/log	defaults (nosuid, noexec, nodev – can be added)
/var/tmp	nosuid, noexec, nodev
/var/log/audit	defaults (nosuid, noexec, nodev – can be added)
Any Removable Media Partition	nosuid, noexec, nodev

Focus should be aligned with the following options being assigned to temporary storage mount points (e.g., `/tmp`, `/var/tmp`, `dev/shm`, removable media):

- `noexec`: does not allow for direct execution of any binaries on the mounted filesystem.
- `nodev`: does not interpret character or block special devices on the file system.
- `nosuid`: does not allow set-user-identifier or set-group-identifier bits to take effect.

To verify what options a mounted filesystem is utilizing, the mount command can be leveraged. It is also possible to include the grep command to limit the output for a particular mount point (Figure 13).

```
mount | grep /tmp
/usr/.tempdisk on /tmp type ext4
(rw,nosuid,nodev,noexec,relatime)
```

FIGURE 13. Verifying the mount options for `/tmp` mount point.

The output from the mount command will display the currently mounted filesystems, the filesystem type the device is mounted as, and the options that are utilized.

If the recommended options are not present for the mount point, then organizations should configure the file system table (`/etc/fstab`) to add the recommended options for the associated mount points.

Note: If the mount points referenced in Table 2 do not exist, evaluate the feasibility to create another partition, mount it, and then apply the recommended options. When modifying an existing folder (e.g., `/var/log`) it is advisable to bring the system to emergency mode by ensuring that auditd is not running.

SELinux

SELinux is a Mandatory Access Control (MAC) security module for Red Hat Enterprise Linux (RHEL) and CentOS. SELinux can control access based on labels assigned to files and processes. Labels are configured in the form of `user:role:type:level`.

To verify if SELinux is installed and configured, the sestatus command can be leveraged. Additionally, the getenforce command can be leveraged to determine the configuration state of SELinux.

SELinux Modes

The configuration of SELinux is managed within the `/etc/selinux/config` file. Within the configuration file, SELinux can be configured based upon the following modes:

- **Enforcing:** SELinux security policies are enforced for each system call. All actions will be logged to `/var/log/audit/audit.log`.

```
# Command to set SeLinux to Enforcing mode
sudo setenforce 1

# Parameters to add to the /etc/selinux/
config file
SELINUX=enforcing
```

FIGURE 14. SELinux enforcement mode.

- **Permissive:** SELinux prints warnings instead of enforcing. All actions will be logged to `/var/log/audit/audit.log`.

```
# Command to set SeLinux to Enforcing mode
sudo setenforce 0

# Parameters to add to the /etc/selinux/
config file
SELINUX=permissive
```

FIGURE 15. SELinux permissive mode.

- **Disabled:** No SELinux policies are loaded. No actions will be logged

```
# Parameters to add to the /etc/selinux/
config file
SELINUX=disabled
```

FIGURE 16. SELinux disabled mode.

Note: A system restart is required to switch between SELinux modes.

If SELinux is **not** already running in "Enforcing" mode, it is recommended to first configure SELinux in "Permissive" mode and ensure that autolabeling is enforced (Figure 17) upon reboot.

```
touch /.autorelabel
```

FIGURE 17. Autolabeling command.

SELinux Policies

An SELinux policy is a set of rules that define the directories, files, and ports that can be accessed by a process or application. SELinux polices can be created either automatically or manually.

Targeted mode within the `/etc/selinux/config` file is the default mode for SELinux. In this mode, SELinux targets only selected processes that exist in configured domains.

SELinux Context

To make access control decisions, SELinux uses "context" to identify the associated resources relevant to an application or process. Context is the collection of security related information assigned to each object (e.g., directory, file, process, port).

```
system_u:object_r:net_conf_t:s0
```

FIGURE 18. SELinux context format.

- The command ls `-Z` will display the SELinux security context files.
- The command ps `-efZ` will display the SELinux security context for running processes.
- The command id `-Z` will display the SELinux security context for users.

SELinux uses four context parameters that are enforced by policies:

- **Users:** Every account, process, or application in Linux can be linked to one "SELinux user" mapping.

  To change a user association to a different SELinux user mapping, the command referenced in Figure 19 can be leveraged:

```
semanage login -a -s <SELinux-User> <account>
```

FIGURE 19. SELinux user context change command.

- **Roles:** Defines what an SELinux user can do with an object in a specified domain or type. To check what domains or types a specific role can access, the `seinfo` command can be leveraged (Figure 20).

```
seinfo -<role>_r -x
```

FIGURE 20. SELinux role association command.

- **Type / Domain:** Logical grouping of objects (type) or processes (domain) to apply permissions.

  To change the SELinux context for objects or processes, the `chcon` command can be leveraged (Figure 21).

```
chcon -t <type-name>_t <file path / name>
```

FIGURE 21. SELinux type modification command example.

- **Sensitivity:** Defines multiple levels of security between level c0 and c3. This context is used only when the SELinux policy type is set to MLS mode (non-default).

SELinux Modules

SELinux uses modules to load permission configurations. To view the list of loaded modules, the command referenced in Figure 22 can be leveraged.

```
sudo semodule -l
```

FIGURE 22. Command to view the list of SELinux loaded modules.

The commands referenced in Figure 23 can be leveraged to control modules.

```
# Disable a module
semodule -d <Module Name>

# Enable a module
semodule -e <Module Name>

# Remove a module
semodule -r <Module Name>
```

FIGURE 23. Commands to manage SELinux loaded modules.

For additional information about SELinux, reference: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/selinux_users_and_administrators_guide/index

AppArmor

AppArmor is another method to apply MAC for Ubuntu and SuSE Linux systems. Unlike SELinux, with AppArmor, MAC rules for applications are applied by file paths instead of within security contexts (labels). AppArmor works by profiling an application to determine what an application needs to access, and the capabilities required to function as part of normal baselined operations.

AppArmor profiles can allow capabilities like network access, raw socket access, and the permission to read, write, or execute files based on matching paths. Profiles (stored in the `/etc/apparmor.d` directory) can also be configured and customized based upon specific permissions required by an application. Profiles can run in "complain mode" or "enforce mode."

- Enforce mode (default setting for the profiles that come with Ubuntu) prevents applications from taking restricted actions.
- Complain mode allows applications to take restricted actions and creates a log entry recording the action within `/var/log/` messages.

To install AppArmor, the command referenced in Figure 24 can be leveraged.

```
sudo apt install apparmor
```

FIGURE 24. Command to install AppArmor

To verify if AppArmor is running and configured, the command referenced in Figure 25 can be leveraged.

```
apparmor_status
```

FIGURE 25. Command to verify if AppArmor is installed and configured.

For any profiles that are not configured for "enforce mode", the commands reference in Figure 26 can be leveraged.

```
# specific profile
sudo aa-enforce /path/to/bin

# all profiles
sudo aa-enforce /etc/apparmor.d/*
```

FIGURE 26. Command to verify if AppArmor is installed and configured.

Note: The `aa-complain` command can be substituted to change the profiles from "enforce mode" to "complain mode".

The `apparmor_parser` command can be used to load a profile into the kernel. It can also be used to reload a currently loaded profile (`-r` option) after modifying it to have the changes take effect (Figure 27).

```
# reload a profile:
sudo apparmor_parser -r /etc/apparmor.d/
profile.name

# reload all profiles:
sudo systemctl reload apparmor.service
```

FIGURE 27. Commands to reload an AppArmor profile.

The `/etc/apparmor.d/disable` directory can be used along with the `apparmor_parser -r` option to disable a profile (Figure 28).

```
sudo ln -s /etc/apparmor.d/<profile.name> /
etc/apparmor.d/disable/
sudo apparmor_parser -r /etc/
apparmor.d/<profile.name>
```

FIGURE 28. Commands to disable an AppArmor profile.

To disable AppArmor and unload the kernel module, the commands referenced in Figure 29 can be leveraged:

```
sudo systemctl stop apparmor.service
sudo update-rc.d -f apparmor remove
```

FIGURE 29. Commands to unload the AppArmor kernel module.

To re-enable AppArmor, the commands referenced in Figure 30 can be leveraged.

```
sudo systemctl start apparmor.service
sudo update-rc.d apparmor defaults
```

FIGURE 30. Commands to re-enable AppArmor.

For additional information about AppArmor, reference: https://ubuntu.com/server/docs/security-apparmor

Configure Iptables to Enforce Local Firewall Rules

Iptables is the local firewall enforcement control for Linux endpoints. Iptables configurations can be leveraged to control both inbound and outbound layer 3 communications in Linux.

Figure 31 contains example commands to configure specific conditions using iptables in Linux.

```
# List iptables rules
sudo iptables -L -v -n

# Flush command to clean-up iptables rules
iptables –flush

# Modify the default chain policy to DROP (default = ACCEPT)
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP
# Note: When DROP is configured for both INPUT and OUTPUT, for every defined communication flow,
two (2) rules (one for incoming and one for outgoing) must be configured.

# Only allow Inbound SSH from a specific subnet
iptables -A INPUT -i eth0 -p tcp -s <XXX.XXX.XXX.XXX/XX> --dport 22 -m state --state
NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT

# Only allow Outbound SSH to a specific subnet
iptables -A OUTPUT -o eth0 -p tcp -d < XXX.XXX.XXX.XXX/XX> --dport 22 -m state --state
NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT

# Block a specific IP address Inbound
iptables -A INPUT -s <XXX.XXX.XXX.XXX> -j DROP

# Block a specific IP address Outbound
iptables -A OUTPUT -d <XXX.XXX.XXX.XXX> -j DROP

# Block a specific IP address and port Outbound
iptables -A OUTPUT -p tcp -d <XXX.XXX.XXX.XXX> --dport <XXXX> -j DROP
```

FIGURE 31. Example iptables configurations.

Once configured, iptables rules need to be saved using either of the commands referenced in Figure 32.

```
/sbin/service iptables save
/etc/init.d/iptables save
```

FIGURE 32. Commands to save iptables rules.

The scope of ports and protocols to configure within iptables will vary based upon the intended use-case of the Linux endpoint. As a best-practice, the goal of an iptables configuration would enforce a default chain policy of DENY, with only allow-list communications configured bi-directionally.

Note: Linux distributions can use different firewall services and firewall management tools. To avoid potential conflicts with the different services, it is recommended to run only one firewall service on a Linux endpoint. Common modern firewall services and firewall management tools are noted below:

- RPM-based Linux distributions such as RHEL, CentOS, Fedora, SUSE and OpenSUSE use firewalld by default. For additional details about firewalld, reference https://firewalld.org/
- Uncomplicated Firewall (ufw) is available by default in Ubuntu distributions. For additional details about ufw, reference https://wiki.ubuntu.com/UncomplicatedFirewall
- Nftables is the default and recommended firewalling framework in Debian 10 and later, and it replaces iptables. For additional details about nftables, reference https://wiki.debian.org/nftables

Disable Unnecessary Services

Linux endpoints should be reviewed and hardened to protect against non-baselined services from being enabled and leveraged. To see a listing of all enabled and running services on a Linux endpoint, the command service—status-all can be leveraged. Any running services will have a "+" symbol associated (Figure 33).

```
[ + ]  acpid
[ - ]  alsa-utils
[ - ]  anacron
[ + ]  apparmor
[ + ]  apport
[ - ]  avahi-daemon
[ + ]  bluetooth
[ - ]  console-setup.sh
[ + ]  cron
[ + ]  cups
[ + ]  cups-browsed
[ + ]  dbus
[ + ]  gdm3
[ - ]  grub-common
[ - ]  hwclock.sh
[ + ]  irqbalance
[ + ]  kerneloops
[ - ]  keyboard-setup.sh
[ + ]  kmod
[ + ]  network-manager
[ + ]  openvpn
[ - ]  plymouth
[ - ]  plymouth-log
[ - ]  pppd-dns
[ + ]  procps
[ - ]  pulseaudio-enable-autospawn
[ - ]  rsync
[ + ]  rsyslog
[ - ]  saned
[ - ]  speech-dispatcher
[ - ]  spice-vdagent
[ + ]  udev
[ + ]  ufw
[ + ]  unattended-upgrades
[ - ]  uuidd
[ + ]  whoopsie
[ - ]  x11-common
```

FIGURE 33. Example output of running services.

To disable a specific service, the command `systemctl`—now disable `<service-name>` can be leveraged.

Common Linux endpoint services that can be abused by adversaries and should be considered in-scope for disabling include:

- avahi-daemon
- NFS
- rpcbind
- rsync
- smb
- telnet

Note: The services listed above can be used legitimately, therefore the business justification for the services to remain enabled should be considered. To reduce the attack surface of a Linux endpoint, if the services are not required for operational purposes, they should be disabled. If the services cannot be disabled, at a minimum, local firewall rules should be configured to restrict the scope of inbound access to Linux endpoints for some of the common protocols and ports noted below.

- NFS (TCP & UDP/2049)
- rpcbind (TCP & UDP/111)
- smb (TCP/445)
- telnet (TCP/22)

NFS Server Hardening

Network File Sharing (NFS) is a protocol that allows for directories and files to be shared with Linux clients over a network. Configuration parameters for an NFS server are stored within the `/etc/exports` file.

Best practices for hardening an NFS server to prevent against unauthorized access to the file and directory contents include:

- If directories must be mounted with the `rw` option (which allows for both read and write requests on the NFS volume), to reduce risks of ransomware or data being overwritten in a malicious manner. ensure that the volumes and directories are not world-writable
- Leverage the `showmount` command to review all exported directories. If directories or paths that should not be exportable are listed, restrictions can be defined within the `/etc/exports` file.

File Integrity Monitoring (FIM)

Modern Linux platforms can leverage the Advanced Intrusion Detection Environment (AIDE) utility to create a database of files to verify their integrity—and to identify any newly created, modified, or removed files. To generate an initial AIDE database to baseline the filesystem, the `aide –init` or `aideinit` commands can be leveraged.

Note: AIDE initialization will check directories and files defined within the /etc/aide.conf file. To include additional directories or files within the database, and to change any parameters, the /etc/aide.conf file can be edited and modified.

Once initialized, AIDE can be executed either manually (`aide–check` or the `aide -c /etc/aide/aide.conf –check` commands) or as part of a scheduled (cron) job within `/etc/crontab`. The output will identify files that have been added, removed, or modified since AIDE was last initiated or ran. Example output is included within Figure 34. For any added, removed, or modified files, the output from AIDE will provide additional context about the directory and file attributes.

```
AIDE found differences between database and
filesystem!!
Verbose level: 6

Summary:
  Total number of entries: 286611
  Added entries:           8
  Removed entries:  4
  Changed entries:  28
```

FIGURE 34. Example AIDE output.

For additional information about AIDE, reference:
https://help.ubuntu.com/community/FileIntegrityAIDE

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/security_guide/sec-using-aide

Credential Hardening

Root Account Hardening

Organizations should not only limit the root account usage for Linux endpoints, but also regularly review and monitor any root account usage in the environment and perform best-practices such as:

- configuring a strong password.
- enforcing a password rotation on a pre-defined interaval.
- setting a unique password for root accounts on all Linux endpoints in the environment.

Identify and Protect Privileged Accounts

Organizations should review and document all accounts configured on all Linux endpoints. All local accounts on a Linux endpoint are listed in the files `/etc/passwd and /etc/sudoers`. Any modifications to content within these files should be reviewed to ensure that all accounts are authorized.

The UID (user identifier) value is used by Linux to identify a user. Any user account that has the UID value set to 0 will have root privileges. Organizations should check and ensure that the only account configured on a Linux system with UID `0` is the default root account. The command referenced in Figure 35 can be used to identify accounts that have a UID value set to `0`.

```
cat /etc/passwd | awk -F: '($3 == 0) { print
$1 }'
```

FIGURE 35. Command to identify accounts with UID 0.

The `sudo` command allows users to run programs with the security privileges of another user. By default, sudo runs commands with superuser privileges. Access to the sudo privilege is managed through the `/etc/sudoers` file. Alternatively, file entries can also be created in the `/etc/sudoers.d` folder to provide sudo access to a user. Organizations should review the `/etc/sudoers` file and `/etc/sudoers.d` folder to identify any groups that may have sudo access. Membership to any groups that have sudo access should be reviewed.

In RHEL, the group "`wheel`" has sudo access as part of the default configuration. The command referenced in Figure 36 can be used to identify all the members of a group (e.g., "wheel") on a RHEL endpoint.

```
grep wheel /etc/group
```

FIGURE 36. Command to list all members of the wheel group.

Strong Password Enforcement

Organizations should consider implementing mechanisms to ensure strong passwords are used for all accounts on Linux endpoints. The "`Pam_cracklib.so`" module can be used to check the strength of passwords.

The command referenced in Figure 37 can be used to install the `libpam_cracklib` module.

```
$ sudo apt install libpam-cracklib
```

FIGURE 37. Command to install the `libpam_cracklib module\`.

After the installation has completed, the `/etc/pam.d/common-password` file will need to be edited to enforce strong password requirements (Figure 38).

The available parameters for editing include:

- **retry=1 :** Prompts user at most 1 time before returning with error. The default is also 1.
- **minlen=15 :** The minimum acceptable size for the new password.
- **ucredit=-1 :** The password must contain at least 1 uppercase characters.
- **lcredit=-1 :** The password must contain at least 1 lowercase characters.
- **dcredit=-12 :** The new password must contain at least 12 digits.
- **ocredit=-12 :** The new password must contain at least 12 symbols.

```
sudo cp /etc/pam.d/common-password /root/
sudo nano /etc/pam.d/common-password

# Edit parameters
password requisite pam_cracklib.so retry=1
minlen=15 ucredit=-1 lcredit=-1 dcredit=-1
ocredit=-1
```

FIGURE 38. Steps to enforce strong password requirements on a Linux endpoint.

The settings referenced in Figure 39 are recommended to enable password lockout for failed authentication attempts, where the example enforces an account lockout after five (5) failed logon attempts.

```
auth required pam_faillock.so preauth audit silent deny=5 unlock_time=900
auth [success=1 default=bad] pam_unix.so
auth [default=die] pam_faillock.so authfail audit deny=5 unlock_time=900
auth sufficient pam_faillock.so authsucc audit deny=5 unlock_time=900
```

FIGURE 39. Enable Lockout for failed password attempts.

The setting referenced in Figure 40 will prevent root account lockout as a result of an account lockout configuration.

```
auth required /lib/security/$ISA/pam_tally.so onerr=fail no_magic_root
```

FIGURE 40. Configuration to protect root accountsa from lockout.

The setting referenced in Figure 41 will enforce that the sha512 hashing algorithm is used as part of the password storage configuration.

```
password sufficient pam_unix.so sha512
```

FIGURE 41. Configure usage of sha512 for password hashing.

Accounts that are inactive for a long period should be audited and disabled on a periodic basis. The `lastlog` command can be used to identify the last login date for a specific account.

Interactive Logon Restrictions

Organizations should configure the shell field in the `/etc/passwd` file to "`/sbin/nologin`" for accounts that don't require interactive logons, which effectively disables shell access for an account (Figure 42).

```
usermod -s /sbin/nologin <user>
```

FIGURE 42. Command to set the shell for a user as `/sbin/nologin` for a specific account.

Auditing and Visibility

System Auditing Configurations

Organizations should review Linux endpoint auditing configurations present in the Linux Auditing System (AuditD) to ensure that events and detections are recorded and available for review and analysis.

Linux Auditing System (Auditd)

AuditD is a feature for Linux endpoints that provides logging of actions such as system calls, file access, authentication, and other security-related events. When installed (`sudo apt-install auditd`) and enabled (`sudo service auditd start`), there are several types of security events that are recorded, such as:

- Modification to audit configurations and audit log files.
- Changes to trusted databases (e.g., `/etc/passwd`).
- Auditing and monitoring `/etc/passwd` for write, append, and read actions (using `auditctl`).
- Attempts to export information out of the system.
- Changes to user authentication mechanisms (e.g., SSH).
- Modifications to user and group associations.
- Login and logout events.
- Unsuccessful unauthorized file access attempts.
- Privileged commands being invoked.
- File deletion events.
- Changes to privileged functions (e.g., sudoers).
- Kernel module loading and unloading.
- Auditing of all privileged functions.
- File mount conditions.
- Locations of public/private keys (within the `.ssh` directory)

While there are many benefits of enabling auditd, there are also some potential performance impacts. Once configured, auditd logs can quickly accumulate on an endpoint, so organizations should have a mechanism in place to forward audit logs to a log aggregation tool or a SIEM. To enhance the performance of auditd, it is recommended to configure the `/etc/audit/auditd.conf` file to allow for the endpoint to have enough time to accumulate the logs to forward off the endpoint, as well as ensuring that logs are written to disk and not to the event buffer to protect against overutilization of the `userspace` (Figure 43).

Figure 43 provides sample `auditd.conf` configuration options that reflect optimized settings for both logging and performance. These settings should be tested on a subset of Linux endpoints to ensure that they are optimized for the current environment.

```
local_events = yes
write_logs = yes
log_file = /var/log/audit/audit.log
log_group = root
log_format = RAW
flush = INCREMENTAL
freq = 50
num_logs = 99
max_log_file = 50
max_log_file_action = ROTATE
priority_boost = 4
disp_qos = lossy
#dispatcher = /sbin/audispd
name_format = NONE
##name = mydomain
space_left = 524
space_left_action = EMAIL
verify_email = yes
action_mail_acct = root
admin_space_left = 256
admin_space_left_action = SUSPEND
disk_full_action = SUSPEND
disk_error_action = SUSPEND
use_libwrap = yes
##tcp_listen_port = XX
tcp_listen_queue = 5
tcp_max_per_addr = 1
##tcp_client_ports = 1024-65535
tcp_client_max_idle = 0
enable_krb5 = no
krb5_principal = auditd
##krb5_key_file = /etc/audit/audit.key
distribute_network = no
```

FIGURE 43. Example auditd configuration settings.

After the `/etc/audit/auditd.conf` file has been configured, the next step to enhance visibility is to add additional rules to the `/etc/audit/rules.d/audit.rules` file. Some examples of events that can be generated by auditd rules include, but are not limited to:

- Reading, configuration, and monitoring of audit tools
- Standard kernel parameters, loading and unloading of modules (including modprobe configuration), and `kexec` usage
- Use of special files (e.g., attached block devices), mount operations, and swap operations
- Cron configurations
- Changes to users, groups, and passwords
- Changes to Sudoers/root privileges, login information
- Network events (e.g., hostname changes and connections)
- Startup scripts and search paths
- Service and system configurations
- Access to sensitive directories and binaries (e.g., /sbin)
- Process ID changes
- Session initiation
- Changes to sensitive access control levels (e.g., chmod >= 500)
- Common reconnaissance, suspicious use of binaries (e.g., Netcat, and code/data/process injections)
- Suspicious file access

The scope and type of rules required for auditing and visibility purposes will be different for each organization. Pre-configured rules for consideration are available from the following organizations:

- Center for Information Security (CIS)[1]
- Security Technical Implementation Guide (STIG)[2]
- Information Systems Security Organization (ISSO)
- Controlled Access Protection Profile (CAPP)[3]
- Labeled Security Protection Profile (LSPP)[4]

Log Execution Timestamps (Shell History)

Linux systems store commands executed by a user in a user-specific hidden "`.bash_history`" file located in each user's home directory. This file can provide important information that is useful when investigating and reviewing a Linux endpoint. A limitation of the default shell history recording on Linux is that the history file does not contain timestamps. The availability of timestamps being recorded within the `.bash_history` file can provide significant contextual information for an investigation.

Organizations should enable timestamps for the `.bash_history` file by configuring a "`HISTTIMEFORMAT`" variable on the endpoint. The file "`.bash_profile`" in a user's home directory can be used to configure this every time the user logs into the endpoint (Figure 44).

```
echo 'export HISTTIMEFORMAT="%d/%m/%y %T "'
>> ~/.bash_profile
```

FIGURE 44. Command to configure $HISTTIMEFORMAT variable in a .bash_profile file.

The default size of the history file is 1,000 lines. This means that any commands that are older than last 1,000 commands will be removed from the history file; therefore, it is recommended to increase the file's length. This can be configured by setting both the `HISTSIZE` and `HISTFILESIZE` variables on a Linux endpoint. The commands referenced in Figure 45 will configure variables in a user's bash profile, increasing the size of the history file to an unlimited value.

```
echo 'export HISTSIZE= ' >> ~/.bash_profile
echo 'export HISTFILESIZE= ' >> ~/.bash_
profile
```

FIGURE 45. Commands to configure unlimited history file size for bash history.

Session Recording

Organizations should consider enabling user session recording. User session recording enables the ability to record and play back user terminal sessions. All the recordings are captured and stored in a text-based format in the system journal. This data can be used for auditing user sessions or performing forensics in case of a security incident.

For additional information for how to enable and configure session recording, reference:
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/recording_sessions/index

http://manpages.ubuntu.com/manpages/bionic/man5/sssd-session-recording.5.html

https://manpages.debian.org/testing/sssd-common/sssd-session-recording.5.en.html

https://fedoraproject.org/wiki/ScreenCasting

1 https://benchmarks.cisecurity.org/tools2/linux/CIS_CentOS_Linux_7_Benchmark_v1.1.0.pdf
2 https://github.com/linux-audit/audit-userspace/blob/master/rules/30-stig.rules
3 https://www.commoncriteriaportal.org/files/ppfiles/capp.pdf
4 https://www.commoncriteriaportal.org/files/ppfiles/lspp.pdf

Conclusion

Linux-focused exploitation attacks can pose a serious threat to organizations. This whitepaper provides practical hardening guidance to protect against common techniques used by threat actors to access and deploy malware or backdoors on Linux endpoints. The guidance provided within this document is based on front-line expertise with helping organizations prepare, contain, eradicate, and recover from incidents where Linux endpoints have been targeted and impacted.

Learn more at **www.mandiant.com**

Mandiant
11951 Freedom Dr, 6th Fl, Reston, VA 20190
(703) 935-1700
833.3MANDIANT (362.6342)
info@mandiant.com

About Mandiant
Since 2004, Mandiant® has been a trusted partner to security-conscious organizations. Today, industry-leading Mandiant threat intelligence and expertise drive dynamic solutions that help organizations develop more effective programs and instill confidence in their cyber readiness.

MANDIANT

TABLE 2. Change log.

Version/Date	Notes
1.0: March 15, 2022	Initial Document