

CYBERPATRIOT MINT 21 ULTRA-DETAILED CHECKLIST
DESIGNED FOR BASH SCRIPT CREATION (BUT ALSO MANUAL USE)

Goal: Every bullet should be something you could eventually automate in a Mint 21 hardening script.

=====

0. GLOBAL SCRIPT SETUP (BEFORE TOUCHING THE SYSTEM)

0.1 Detect distro (guardrails in your script)

```
# Script idea:  
. /etc/os-release  
if [ "$NAME" != "Linux Mint" ] || [ "$VERSION_ID" != "21" ]; then  
    echo "This script is meant for Linux Mint 21. Exiting."  
    exit 1  
fi
```

0.2 Initialize logging

- Create a log directory and log file for your script:

```
LOG_DIR="/root/cp-log"  
LOG_FILE="$LOG_DIR/mint21-$(date +%F-%H%M%S).log"  
sudo mkdir -p "$LOG_DIR"  
sudo touch "$LOG_FILE"
```
- Create a logging helper in your script:

```
log() { echo "[$(date +%F-%T)] $*" | sudo tee -a "$LOG_FILE"; }
```

0.3 Update locate DB early

- Many checks will use `locate` for speed:

```
sudo updatedb
```
- Put this near the beginning of your script.

0.4 Respect the CyberPatriot order of operations

- From the basic Linux security guide, recommended order: README → Forensics → Users → Password Policy → Local Security Policy → Services → Auto Updates → Software Updates → Browser → Prohibited Software/Media → OS Updates.
[oai_citation:0‡Basic-Linux-Security-Reading.pdf](sediment://file_00000000b2c871f581527f7efffe3079)
 - Your script should NOT blindly delete forensics-related files; keep a “FORENSIC_MODE=1” flag so you can choose to skip destructive actions until after you’re done answering questions.

Example guard:

```
FORENSIC_MODE=1  
# Later, once forensics are done, set to 0
```

```
# FORENSIC_MODE=0
```

```
=====
1. FORENSICS SUPPORT (MOSTLY MANUAL, BUT SCRIPT CAN HELP)
=====
```

You can't fully automate forensics (you don't know the exact questions), but scripts can:

1.1 Snapshot key info for manual analysis (non-destructive)

- At script start, capture process and network state:

```
log "Capturing initial process list and listeners..."  
ps -ef | sudo tee -a "$LOG_FILE"  
ss -tulnp | sudo tee -a "$LOG_FILE"
```

- This helps find python backdoors like `/usr/share/zod/kneelB4zod.py` listening on port 1337.

[oai_citation:1‡CP-Mint21-Training2-Answer-Key.pdf](sediment://file_0000000269c722faa8688903b5b2a62)

1.2 Install forensics tools if missing (tshark)

- For pcap questions (like Mint21 Training & similar):

```
sudo apt-get update  
sudo apt-get install -y tshark  
log "Installed tshark for network forensics."
```

[oai_citation:2‡CP-Mint21-Training2-Answer-Key.pdf](sediment://file_0000000269c722faa8688903b5b2a62)

1.3 Provide helper commands for pcap analysis (printed to screen)

- Your script can echo hints:

```
log "For FTP pcap passwords, try:"  
log " cd ~/Desktop"  
log " tshark -q -z follow,tcp,ascii,0 -r ftp_capture.pcap | grep PASS"  
log " tshark -q -z follow,tcp,ascii,21 -r ftp_capture.pcap | grep PASS"
```

[oai_citation:3‡CP-Mint21-Training2-Answer-Key.pdf](sediment://file_0000000269c722faa8688903b5b2a62)

1.4 Helper: locate mp3 paths for forensics

- Sometimes forensics ask for absolute directory path of MP3s.

[oai_citation:4‡CP-Ubuntu22-Training2-Answer-Key.pdf](sediment://file_0000000683c71fb943e9a804acaaa0c)

```
log "Searching for .mp3 files (for forensics)..."  
locate "*.mp3" | sudo tee -a "$LOG_FILE"
```

1.5 IMPORTANT: Don't delete anything in forensic paths until questions answered

- Guard in script for destructive actions:

```
if [ "$FORENSIC_MODE" -eq 1 ]; then
```

```

    log "FORENSIC_MODE=1: Skipping destructive actions (removal of
users/files/software)."
    # exit or skip these sections
fi
```

=====

2. USER & GROUP MANAGEMENT (USER AUDITING CATEGORY)

=====

Mint 21 CP rounds consistently score:

- Removing unauthorized users, fixing admin rights, creating required users, forcing password change at next login.

2.1 Script variables: define allowed and admin users

- In your script, define arrays:

```
ALLOWED_USERS=("root" "benjamin" "kbennett" "mross" "edarby") # example; adjust per
README
```

```
ADMIN_USERS=("benjamin" "edarby") # example; adjust per README
```

2.2 Enumerate all local accounts and remove unauthorized ones

- In script:

```
log "Checking for unauthorized users..."
ALL_USERS=$(cut -d: -f1 /etc/passwd)
for u in $ALL_USERS; do
    # Skip system accounts (UID < 1000)
    UID=$(id -u "$u" 2>/dev/null || echo 0)
    if [ "$UID" -ge 1000 ] && [ "$u" != "$USER" ]; then
        IS_ALLOWED=0
        for a in "${ALLOWED_USERS[@]}"; do
            [ "$u" = "$a" ] && IS_ALLOWED=1 && break
        done
        if [ "$IS_ALLOWED" -eq 0 ]; then
            log "Removing unauthorized user: $u"
            sudo deluser --remove-home "$u"
        fi
    fi
done
```

- This covers vulnerabilities like “Removed unauthorized user ttanner”, “Removed unauthorized user cdennis”, etc. [oai_citation:5‡CP 18 r1 Img Ans and Vulns.pdf](sediment://file_0000000f4e0722fb7fc097d31b24d68)

2.3 Ensure required users exist

- Script stub:

```
ensure_user() {
```

```

local u="$1"
if ! id "$u" >/dev/null 2>&1; then
    log "Creating missing user: $u"
    sudo adduser --gecos "" "$u"
fi
}
for u in "${ALLOWED_USERS[@]}"; do
    [ "$u" = "root" ] && continue
    ensure_user "$u"
done
- Matches actions like "Created user account mross" and "Created new administrator account for edarby".

```

2.4 Fix admin vs standard account membership

- Let `ADMIN_GROUP="sudo"` (Mint default).
- Script to enforce admin list:

```

log "Fixing admin (sudo) group membership..."
CURRENT_ADMIN=$(/bin/getent group "$ADMIN_GROUP" | cut -d: -f4 | tr ',' '')
# Remove users who shouldn't be admin
for u in $CURRENT_ADMIN; do
    KEEP=0
    for a in "${ADMIN_USERS[@]}"; do
        [ "$u" = "$a" ] && KEEP=1 && break
    done
    if [ "$KEEP" -eq 0 ]; then
        log "Removing $u from $ADMIN_GROUP..."
        sudo gpasswd -d "$u" "$ADMIN_GROUP"
    fi
done
# Add required admins
for a in "${ADMIN_USERS[@]}"; do
    if id "$a" >/dev/null 2>&1; then
        log "Adding $a to $ADMIN_GROUP..."
        sudo usermod -aG "$ADMIN_GROUP" "$a"
    fi
done

```

- This addresses "User kbennett is not an administrator" etc.

2.5 Force specific users to change password at next login

- For users like `mross` or `edarby` that must change password:

```

FORCE_CHANGE_USERS=("mross" "edarby")
for u in "${FORCE_CHANGE_USERS[@]}"; do
    if id "$u" >/dev/null 2>&1; then
        log "Forcing $u to change password at next login..."
    fi
done

```

```
    sudo chage -d 0 "$u"
  fi
done
```

3. PASSWORD POLICY (ACCOUNT POLICY CATEGORY)

Mint 21 Mint CP rounds score:

- Minimum password age
- Minimum password length, etc.

3.1 Enforce system-wide password aging in /etc/login.defs

- Script:

```
log "Configuring password aging in /etc/login.defs..."
sudo sed -i 's/^PASS_MAX_DAYS.*$/PASS_MAX_DAYS 90/' /etc/login.defs
sudo sed -i 's/^PASS_MIN_DAYS.*$/PASS_MIN_DAYS 10/' /etc/login.defs
sudo sed -i 's/^PASS_WARN_AGE.*$/PASS_WARN_AGE 7/' /etc/login.defs
```

3.2 Apply aging to existing human users

- Identify UIDs ≥ 1000:

```
USERS_1000=$(awk -F: '$3 >= 1000 {print $1}' /etc/passwd)
for u in $USERS_1000; do
  log "Applying aging to $u..."
  sudo chage -M 90 -m 10 -W 7 "$u"
done
```

3.3 Enforce minimum password length via PAM

- Mint 21 uses `/etc/pam.d/common-password`.
- Script approach (append or modify pam_pwquality or pam_unix):

```
FILE="/etc/pam.d/common-password"
log "Configuring minimum password length in $FILE..."
# Ensure pam_pwquality configured (if present)
if grep -q "pam_pwquality.so" "$FILE"; then
  sudo sed -i 's/^(pam_pwquality.so.*$)\$/\1 minlen=12/' "$FILE"
fi
# Enforce remember for pam_unix
if grep -q "pam_unix.so" "$FILE"; then
  sudo sed -i 's/^(pam_unix.so.*$)\$/\1 remember=5/' "$FILE"
fi
```
- This matches “A minimum password length is required” in Mint21 R2. [oai_citation:6‡CP 18 r2 Img Ans & Vulns.pdf](sediment://file_00000000c3f871f599b22dd45098cacd)

4. ACCOUNT LOCKOUT / AUTH POLICY (ACCOUNT POLICY)

Not always explicitly scored on Mint, but good practice and scriptable.

4.1 Configure faillock (if available) or pam_tally2

- Example snippet for faillock in `/etc/security/faillock.conf` (manually or via script using `sed`/`tee`):

```
cat << 'EOF' | sudo tee /etc/security/faillock.conf
deny = 5
unlock_time = 900
fail_interval = 900
audit
EOF
```

- Then ensure `auth` stack includes `pam_faillock.so` before `pam_unix.so` in `/etc/pam.d/common-auth` .

4.2 Ensure no null passwords

- In `/etc/pam.d/common-auth` :
- Remove `nullok` option from any `pam_unix.so` line:
sudo sed -i 's/\(pam_unix.so.*\)\nullok\1/g' /etc/pam.d/common-auth

5. GUEST ACCOUNT / DISPLAY MANAGER (UNCATEGORIZED OS SETTINGS)

Mint 21 uses LightDM; CP R2 Mint21 mentions “Guest account is disabled”. [oai_citation:7‡CP 18 r2 Img Ans & Vulns.pdf](sediment://file_0000000c3f871f599b22dd45098cacd)

5.1 Disable guest sessions & autologin in LightDM

- Script:
LIGHTDM_CONF="/etc/lightdm/lightdm.conf"
log "Hardening LightDM guest and autologin..."
if [! -f "\$LIGHTDM_CONF"]; then
 sudo touch "\$LIGHTDM_CONF"
fi
sudo sed -i '/^Seat:*/d' "\$LIGHTDM_CONF"
cat << 'EOF' | sudo tee -a "\$LIGHTDM_CONF"

```
[Seat:*]  
greeter-session=lightdm-gtk-greeter  
allow-guest=false  
autologin-user=  
EOF
```

6. NETWORK KERNEL SETTINGS (DEFENSIVE COUNTERMEASURES)

Mint21 R2 explicitly scores enabling IPv4 TCP SYN cookies. [oai_citation:8‡CP 18 r2 Img Ans & Vulns.pdf](sediment://file_0000000c3f871f599b22dd45098cacd)

6.1 Enable TCP SYN cookies

- Script:

```
SYSCTL_CONF="/etc/sysctl.conf"
log "Enabling IPv4 TCP SYN cookies..."
if grep -q "net.ipv4.tcp_syncookies" "$SYSCTL_CONF"; then
    sudo sed -i 's/^net.ipv4.tcp_syncookies.*$/net.ipv4.tcp_syncookies = 1/' \
"$SYSCTL_CONF"
else
    echo "net.ipv4.tcp_syncookies = 1" | sudo tee -a "$SYSCTL_CONF"
fi
sudo sysctl -p "$SYSCTL_CONF"
```

7. FIREWALL (UFW) (DEFENSIVE COUNTERMEASURES)

Mint 21 CP rounds: “Uncomplicated Firewall (UFW) protection has been enabled.”

7.1 Enable UFW with sensible defaults

- Script:

```
log "Configuring UFW..."
sudo ufw --force reset
sudo ufw default deny incoming
sudo ufw default allow outgoing
# Allow needed services (example; adjust per README)
sudo ufw allow 22/tcp # SSH, if required
sudo ufw allow 21/tcp # FTP, if required
sudo ufw --force enable
sudo ufw status verbose | sudo tee -a "$LOG_FILE"
```

8. SERVICE AUDITING (SERVICE AUDITING CATEGORY)

Mint 21 CP rounds score:

- Apache2 disabled/removed in Round 1 when not required. [oai_citation:9‡CP 18 r1 Img Ans and Vulns.pdf](sediment://file_0000000f4e0722fb7fc097d31b24d68)

- OpenSSH disabled/removed in R2 (but enabled & updated in R1), vsftpd updated, FTP SSL, etc.

You **cannot** script one universal rule for web/SSH/FTP for ALL Mint21 images because requirements differ. Instead, structure script for “profiles” based on README.

8.1 Script profiles (per-image service policy)

- Define at top of script:

```
PROFILE="$1" # e.g., "r1", "r2", "training2"
# Fallback default
[ -z "$PROFILE" ] && PROFILE="generic"
```

8.2 Service management helpers

- Functions:

```
stop_and_disable() {
    local svc="$1"
    if systemctl list-unit-files | grep -q "^$svc"; then
        log "Stopping and disabling $svc..."
        sudo systemctl stop "$svc" 2>/dev/null || true
        sudo systemctl disable "$svc" 2>/dev/null || true
    fi
}
ensure_running() {
    local svc="$1"
    if systemctl list-unit-files | grep -q "^$svc"; then
        log "Enabling and starting $svc..."
        sudo systemctl enable "$svc"
        sudo systemctl start "$svc"
    fi
}
```

8.3 Web server handling (apache2, nginx)

- For Mint21 R1, apache2 is disabled/removed. For R2, vsftpd and FTP are critical; OpenSSH may be disabled.

- Example generic script logic:

```
if [ "$PROFILE" = "r1" ]; then
    stop_and_disable "apache2.service"
    log "Removing apache2..."
    sudo apt-get purge -y apache2 apache2-bin apache2-data || true
fi
```

8.4 SSH service handling

- R1: OpenSSH updated & used; R2: OpenSSH disabled/removed.
- Script logic:

```

if [ "$PROFILE" = "r1" ]; then
    ensure_running "ssh.service"
    log "Updating OpenSSH..."
    sudo apt-get install --only-upgrade -y openssh-server
elif [ "$PROFILE" = "r2" ]; then
    stop_and_disable "ssh.service"
    log "OpenSSH disabled/removed for profile r2."
fi

```

8.5 FTP (vsftpd) and FTP root permissions

- Mint21 R2: vsftpd updated and FTP root permissions fixed; SSL required.
 - Script snippet for R2:
- ```

if ["$PROFILE" = "r2"]; then
 ensure_running "vsftpd.service"
 log "Updating vsftpd..."
 sudo apt-get install --only-upgrade -y vsftpd
 # Fix FTP root permissions (example path; adjust per image)
 FTP_ROOT="/srv/ftp"
 if [-d "$FTP_ROOT"]; then
 log "Fixing FTP root permissions on $FTP_ROOT..."
 sudo chown root:root "$FTP_ROOT"
 sudo chmod 755 "$FTP_ROOT"
 fi
fi

```

---

## 9. PROHIBITED SOFTWARE (UNWANTED SOFTWARE CATEGORY)

---

Mint21 rounds:

- Remove games and hacking/P2P tools like aisleriot, ophcrack, aMule, Wireshark, Zangband.

### 9.1 Define prohibited packages array

- At top of script:
- ```

PROHIBITED_PKGS_COMMON=("aisleriot" "ophcrack")
PROHIBITED_PKGS_R2=("aMule" "wireshark" "zangband")

```

9.2 Removal loop

- Script:

```

remove_pkgs() {
    for pkg in "$@"; do
        if dpkg -l | grep -q "^ii $pkg "; then
            log "Removing prohibited package: $pkg"
            sudo apt-get purge -y "$pkg"
    done
}

```

```

        fi
    done
}
# Apply:
remove_pkgs "${PROHIBITED_PKGS_COMMON[@]}"
if [ "$PROFILE" = "r2" ]; then
    remove_pkgs "${PROHIBITED_PKGS_R2[@]}"
fi
=====
```

10. PROHIBITED FILES (MP3, MEDIA) (PROHIBITED FILES CATEGORY)

Mint21 CP rounds: remove mp3s in user directories.

10.1 Locate MP3s (and optionally other media) and log them

- Script:

```

log "Locating mp3 files..."
MP3_LIST=$(locate '*.mp3' || true)
echo "$MP3_LIST" | sudo tee -a "$LOG_FILE"
```

10.2 Remove non-work-related MP3s

- Basic approach: only automatically delete in user Music dirs (safer than global):

```

for u in $(awk -F: '$3 >= 1000 {print $1}' /etc/passwd); do
    HOME_DIR=$(eval echo "~$u")
    if [ -d "$HOME_DIR/Music" ]; then
        log "Removing mp3 files from $HOME_DIR/Music..."
        sudo find "$HOME_DIR/Music" -type f -name '*.mp3' -delete
    fi
done
=====
```

11. ANTIVIRUS (CLAMAV) (DEFENSIVE COUNTERMEASURES)

Basic Linux checklist recommends clamav.

[oai_citation:10#Basic-Linux-Security-Checklist.pdf](sediment://file_00000000778c722f93e1da8088b9f194)

11.1 Install & update ClamAV

- Script:

```

log "Installing ClamAV..."
sudo apt-get update
sudo apt-get install -y clamav
```

```
log "Updating ClamAV database..."  
sudo freshclam
```

11.2 Run a recursive scan (optional in competition due to time)

- You can choose to:
 - Scan only `/home` to save time:

```
log "Running ClamAV scan on /home..."  
sudo clamscan -i -r --remove=yes /home | sudo tee -a "$LOG_FILE"
```

12. OS & APPLICATION UPDATES (OS & APP UPDATES CATEGORIES)

Mint21 R1 & R2:

- “The system refreshes the list of updates automatically”, “Install updates from important security updates”, “The update manager installs updates automatically”, and updates for Chromium, OpenSSH, systemd, vsftpd.

12.1 Configure APT Periodic

- Script:

```
CONF="/etc/apt/apt.conf.d/10periodic"  
log "Configuring APT periodic updates..."  
sudo tee "$CONF" >/dev/null << 'EOF'  
APT::Periodic::Update-Package-Lists "1";  
APT::Periodic::Download-Upgradeable-Packages "1";  
APT::Periodic::AutocleanInterval "7";  
EOF
```

12.2 CLI: update & upgrade system

- Script:

```
log "Running apt-get update & dist-upgrade..."  
sudo apt-get update  
sudo apt-get dist-upgrade -y
```

12.3 Specific app updates

- Chromium:

```
sudo apt-get install --only-upgrade -y chromium-browser chromium
```
- systemd:

```
sudo apt-get install --only-upgrade -y systemd
```
- OpenSSH (if profile wants it enabled):

```
sudo apt-get install --only-upgrade -y openssh-server
```
- vsftpd (FTP image):

```
sudo apt-get install --only-upgrade -y vsftpd
```
- Log each:

```
log "Updated Chromium / systemd / OpenSSH / vsftpd as applicable."
```

=====

13. SSH CONFIG HARDENING (APPLICATION SECURITY)

=====

Mint21 R1: "SSH root login has been disabled". [oai_citation:11‡CP 18 r1 Img Ans and Vulns.pdf](sediment://file_0000000f4e0722fb7fc097d31b24d68)

13.1 Disable root SSH login in /etc/ssh/sshd_config

- Script:

```
SSHD_CONF="/etc/ssh/sshd_config"
log "Hardening SSH config..."
sudo sed -i 's/^#?PermitRootLogin.*/PermitRootLogin no/' "$SSHD_CONF"
# (optional) disable X11 forwarding for extra hardening
sudo sed -i 's/^#?X11Forwarding.*/X11Forwarding no/' "$SSHD_CONF"
sudo systemctl restart ssh || sudo systemctl restart sshd || true
```

=====

14. FILE PERMISSIONS & SUID CLEANUP

=====

14.1 Secure /etc/shadow

- Script:

```
log "Securing /etc/shadow permissions..."
sudo chown root:shadow /etc/shadow
sudo chmod 640 /etc/shadow
```

14.2 Remove SUID from dangerous binaries (if present)

- Script:

```
log "Checking SUID binaries..."
SUID_BINARIES=$(find / -perm -4000 -type f -xdev 2>/dev/null)
echo "$SUID_BINARIES" | sudo tee -a "$LOG_FILE"
# Example: remove SUID from find or other suspicious tools
for b in /usr/bin/find /usr/bin/vim.tiny; do
    if [ -u "$b" ]; then
        log "Removing SUID bit from $b"
        sudo chmod u-s "$b"
    fi
done
```

=====

15. BROWSER SECURITY

=====

15.1 Chromium basic hardening (script-friendly hints)

- Some settings are GUI only, but you can:
- Remove suspicious extensions by deleting directories from Chromium profile if known.
- Pre-create a secure preferences JSON if you know exact path.
- Minimum:
log "Manual step: secure Chromium (disable insecure extensions, enable HTTPS-only, etc.)."

16. FINAL VALIDATION & REPORT

16.1 Final check: listeners & services

- Script:
log "Final ss -tulnp output:"
ss -tulnp | sudo tee -a "\$LOG_FILE"

16.2 Final check: users

- Script:
log "Final user list:"
cut -d: -f1 /etc/passwd | sudo tee -a "\$LOG_FILE"

16.3 Final check: prohibited packages

- Script:
log "Final check for prohibited packages..."
dpkg -l | egrep "aisleriot|ophcrack|aMule|wireshark|zangband" | sudo tee -a "\$LOG_FILE"

16.4 Final check: MP3

- Script:
log "Re-checking mp3 files..."
locate '*.mp3' | sudo tee -a "\$LOG_FILE"

16.5 Wrap-up

- Have the script print a summary:
log "Mint 21 hardening script completed. Review \$LOG_FILE for details."
echo ">> Manual tasks left: forensics answers, GUI-only browser settings, any README-specific stuff."

17. SCRIPTING STRATEGY REMINDER

- Keep **all image-specific stuff** (like required users, admin list, profile: r1 vs r2) at the top in config variables.
- Make everything else generic and reusable.
- Use `log` everywhere so you can prove what your script did if you have to restart the VM.
- Always read README and tweak the config section **before** running the script in a new image.

Use this checklist as your blueprint. When you start writing the ` `.sh` script, basically turn each section into functions in a single file, controlled by flags like ` `PROFILE` , ` `FORENSIC_MODE` , and lists of allowed/admin users and prohibited packages.