Opsek

Hardening Scripts

Security Assessment Report

Version: 2.0

February, 2026

Contents

Introduction

Sigma Prime was commercially engaged to perform a time-boxed security review of the Opsek components in scope. The review focused solely on the security aspects of these components, though general recommendations and informational comments are also provided.

Disclaimer

Sigma Prime makes all effort but holds no responsibility for the findings of this security review. Sigma Prime does not provide any guarantees relating to the function of the components in scope. Sigma Prime makes no judgements on, or provides any security review, regarding the underlying business model or the individuals involved in the project.

Document Structure

The first section provides an overview of the functionality of the Opsek components contained within the scope of the security review. A summary followed by a detailed review of the discovered vulnerabilities is then given which assigns each vulnerability a severity rating (see Vulnerability Severity Classification), an *open/closed/resolved* status and a recommendation. Additionally, findings which do not have direct security implications (but are potentially of interest) are marked as *informational*.

The appendix provides additional documentation, including the severity matrix used to classify vulnerabilities within the Opsek components in scope.

Overview

Opsek is a team of experienced Web2 and Web3 Security Researchers combining their forces to secure Web3 Companies, Employees and Users. Opsek offers services such as threat modeling, security awareness training and operational security auditing.

This engagement provided a security assessment of publicly accessible hardening scripts developed by Opsek, evaluating the functionality, security posture of configurations and alignment with best practices.

Security Assessment Summary

Scope

The review was conducted on the files hosted on the Opsek/OSs-security repository. The scope of this time-boxed review was strictly limited to files at commit 481bc29.

Note: third party libraries and dependencies were excluded from the scope of this assessment.

Approach

Assessment of hardening scripts across the three operating systems relied primarily on manual verification to confirm that hardening execution functioned as intended, and does not reduce the security posture of the target system. Research was performed to facilitate additional recommendations for improvements to the hardening script.

Tools used to support review of hardening scripts include:

- ShellCheck: `https://github.com/koalaman/shellcheck`
- PSScriptAnalyzer: `https://github.com/PowerShell/PSScriptAnalyzer`

Coverage Limitations

Due to the time-boxed nature of this review, all documented vulnerabilities reflect best effort within the allotted, limited engagement time. As such, Sigma Prime recommends to further investigate areas of the code, and any related functionality, where majority of critical and high risk vulnerabilities were identified.

Findings Summary

The testing team identified a total of 23 issues during this assessment. Categorised by their severity:

- Low: 15 issues.
- Informational: 8 issues.

Detailed Findings

This section provides a detailed description of the vulnerabilities identified within the Opsek components in scope. Each vulnerability has a severity classification which is determined from the likelihood and impact of each issue by the matrix given in the Appendix: Vulnerability Severity Classification.

A number of additional properties of the components, including optimisations, are also described in this section and are labelled as “informational”.

Each vulnerability is also assigned a **status**:

- ***Open:*** the issue has not been addressed by the project team.
- ***Resolved:*** the issue was acknowledged by the project team and updates to the affected components(s) have been made to mitigate the related risk.
- ***Closed:*** the issue was acknowledged by the project team but no further actions have been taken.

Summary of Findings

ID	Description	Severity	Status
OPK-01	Mishandled DNS Configuration In Windows Hardening Script	Low	Resolved
OPK-02	Inconsistent Backup Of Configuration Files On MacOS	Low	Resolved
OPK-03	Fragile Restore Functionality In MacOS Backup Script	Low	Resolved
OPK-04	Flawed Backup Behaviour On MacOS Hardening Scripts	Low	Resolved
OPK-05	Unnecessary Install Of Brew Package Manager	Low	Resolved
OPK-06	Cron Configuration Deleted Without Backup	Low	Resolved
OPK-07	Fragile Update Configuration	Low	Resolved
OPK-08	Filesystem Remount Not Persistent	Low	Resolved
OPK-09	Root Account Shell Change Breaks Login	Low	Resolved
OPK-10	Flawed Firewall Configuration	Low	Resolved
OPK-11	Flawed SSH Port Randomisation	Low	Resolved
OPK-12	SSH Service Force Started	Low	Resolved
OPK-13	Flawed Backup Behaviour On Linux Hardening Scripts	Low	Resolved
OPK-14	Fragile Sudoers File Manipulation	Low	Resolved
OPK-15	Inconsistently Defined Harden_Profile Configurations	Low	Resolved
OPK-16	Insecure Operational Practice Recommended By Windows Hardening Script Guide	Informational	Resolved
OPK-17	Windows Hardening Script Improvement Suggestions	Informational	Resolved
OPK-18	Suggested Alternative Hardening Mechanism For MacOS	Informational	Resolved
OPK-19	MacOS Hardening Script Specifies Support For Outdated Operating System	Informational	Resolved
OPK-20	Linux Hardening Script Improvement Opportunities	Informational	Closed
OPK-21	Aggressive Disable Of System Services	Informational	Resolved
OPK-22	Inconsistent Dry Run Behaviour	Informational	Resolved
OPK-23	Sudo Configuration Improvement Opportunities	Informational	Closed

OPK-01	Mishandled DNS Configuration In Windows Hardening Script		
Asset	`windows/hardening-windows.ps1, windows/README.md`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Low	Likelihood: Low

Description

The Windows hardening script has an optional operation to set a secure DNS entry, however the function that performs this operation is a loop that iterates through each network interface object and sets the DNS server option for every interface.

While this might be acceptable and expected for physical ethernet network interfaces, this function also applies to other types of interfaces such as tunnel adapters for VPNs, bluetooth network connections and the virtual loopback network adapter which may have unintended consequences on other system components and VPN functionality.

In addition, the `README.md` file states that the DNS server configuration will be set to 8.8.8.8, but the function will set the DNS server to 9.9.9.9.

```
Invoke-Safe "→ Setting secure DNS to 9.9.9.9..." {
    Get-DnsClientServerAddress | ForEach-Object {
        Set-DnsClientServerAddress -InterfaceIndex $_.InterfaceIndex -ServerAddresses ("9.9.9.9")
    }
}
```

Hardening script iterates through every interface and sets the DNS server address to 9.9.9.9.

Recommendations

First, detect the number of interfaces that may be valid targets of this action and prompt the user prior to setting the DNS server for the interface.

Upon detection of a tunnel network interface, advise the user that this may impact name resolution if the VPN target is an internal network.

Modify the `README.md` and `hardening-windows.ps1` files so that both the documented configurations and the configurations that are actually performed are the same.

Resolution

The recommendation has been implemented in PR #30. The following fixes were applied:

- The script documentation has been changed to accurately describe the DNS configuration made.
- The script now loops through each detected interface and prompts the user for confirmation on whether DNS should be statically configured. In addition, a detection has been implemented for non-standard network interfaces to alert to the user the potential impacts of this configuration.

OPK-02	Inconsistent Backup Of Configuration Files On MacOS		
Asset	`macos/*.sh`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Low	Likelihood: Low

Description

The macOS hardening script inconsistently performs back up operations of files that are modified throughout the script, and fails to back up files in some functions where files are overwritten.

For example, in `macos/modules/cis/users.sh` the function `disable_automatic_login` performs the `backup_file` operation on the file `com.apple.loginwindow.plist` prior to editing it.

However, in the following modules files are edited and a backup operation is not performed:

- `macos/modules/cis/network.sh` - `show_bluetooth_status` function makes changes to `com.apple.systemuiserver.plist` file without first backing up the file
- `macos/modules/internals/wifi.sh` - `disable_wifi` does not backup the `com.apple.airport.preferences.plist` file

```
disable_wifi() {
    info "OPSEK - Disabling Wi-Fi"

    # Get all Wi-Fi interfaces
    local wifi_interfaces=$(networksetup -listallhardwareports | awk '/Wi-Fi|AirPort/ {getline; print $2}')

    for interface in $wifi_interfaces; do
        if [[ -n "$interface" ]]; then
            execute "networksetup -setairportpower '$interface' off"
            success "Wi-Fi disabled on interface: $interface"
        fi
    done

    # Disable Wi-Fi menu bar icon
    execute "defaults write com.apple.systemuiserver 'NSStatusItem Visible com.apple.menu.airport' -bool false"

    # Prevent automatic joining of Wi-Fi networks
    execute "defaults write /Library/Preferences/SystemConfiguration/com.apple.airport.preferences DisableAssociation -bool true"

    # Disable Wi-Fi networking entirely
    execute "defaults write /Library/Preferences/SystemConfiguration/com.apple.airport.preferences AllowEnable -bool false"

    success "Wi-Fi completely disabled"
}
```

Note, this finding is systemic across the script, and there are many instances of files being modified without first having a backup operation performed.

Recommendations

Ensure a file backup operation has been performed prior to any filesystem write operation.

Resolution

The recommendation has been implemented in PR #21. Backups are now performed on files directly edited during hardening operations.

OPK-03	Fragile Restore Functionality In MacOS Backup Script		
Asset	`macos/utils/backup.sh`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Low	Likelihood: Low

Description

A rollback script is generated by the macOS hardening scripts that contain two flaws:

- The restore file will attempt to use the base restore of `~/Library/Preferences` for plist files that do not contain `com.apple.*`, however the base restore directory is stored as a variable with double quotes which will not be expanded to the home path by the `cp` command under Bash and will fail.

```
# Restore plist files
find "$BACKUP_DIR" -name "*.plist" | while read -r plist; do
    filename=$(basename "$plist")
    if [[ "$filename" == "com.apple."* ]]; then
        target_dir="/Library/Preferences"
    else
        target_dir="~/Library/Preferences"
    fi

    echo "Restoring $filename to $target_dir"
    cp "$plist" "$target_dir/" 2>/dev/null || echo "Failed to restore $filename"
done
```

- The script executes a restore of the `/etc/sudoers` file without first validating that the file it is restoring from is safe. Should there be an error parsing the file, this would prevent privileged access to the system by all users and likely require a re-install of macOS to rectify.

```
if [[ -f "$BACKUP_DIR/etc/sudoers" ]]; then
    echo "Restoring sudoers"
    cp "$BACKUP_DIR/etc/sudoers" "/etc/sudoers" 2>/dev/null || true
fi
```

Recommendations

Perform the following actions to resolve this vulnerability:

- If it is not possible to remove the double quotes identified, it will be best to pre-expand the users home folder and then concatenating that output value with `Library/Preferences` to generate the full absolute path, or utilise the `$HOME` variable. However, should the script be run under sudo it must be determined first if either of these mechanism will resolve the home folder of the current user or the root user
- Validate the source sudoers file using visudo prior to copying it over the current sudoers file. This can be performed with the command `visudo -c -f /path/to/sudoers`

Resolution

The recommendation has been implemented in PR #20. The ~character has been replaced with the `$HOME` variable and a check is now performed with the `visudo -c -f` command to validate the sudoers file prior to restoring it to the system.

OPK-04	Flawed Backup Behaviour On MacOS Hardening Scripts		
Asset	`macos/utils/common.sh`, `macos/modules/cis/users.sh`, `macos/modules/cis/permissions.sh`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Medium	Likelihood: Low

Description

The backup function defined in the macOS hardening script overwrites the existing backup file whenever the function is called multiple times for same file. When this function is called multiple times, the original backup of the file is overwritten.

This behaviour was identified under the following functions:

- `permissions.sh` & `users.sh` - multiple calls result in overwriting the original backup file for `com.apple.loginwindow.plist`
- `system.sh` - multiple calls result in overwriting the original backup file for `/etc/security/audit-control`

```
# Backup file before modification
backup_file() {
    local file="$1"

    if [[ ! -f "$file" ]]; then
        debug "File does not exist, skipping backup: $file"
        return 0
    fi

    if [[ "$DRY_RUN" == true ]]; then
        debug "[DRY-RUN] Would backup: $file"
        return 0
    fi

    local backup_path="$CURRENT_BACKUP$(dirname "$file")"
    execute "mkdir -p '$backup_path'"
    execute "cp -p '$file' '$backup_path/'"
    debug "Backed up: $file"
}
```

The backup function fails to check if a file backup of a file has already been created before overwriting it with a new configuration.

Recommendations

Perform a check to determine if a backup file already exists. As the backup directory is created with a timestamp in the name, repeated runs of the hardening script will not conflict with each other.

Due to this, a simple check if the backup file already exists will preserve the original file backup without significant alteration to the caller functions.

Resolution

The recommendation has been implemented in PR #20. The `backup_file` function now performs a check to determine if the backup file already exists to prevent the script from overwriting the original backup file.

OPK-05	Unnecessary Install Of Brew Package Manager		
Asset	`macos/install.sh`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Low	Likelihood: Low

Description

The macOS hardening script installs the Brew package manager, however this package manager is not currently required by the script.

Brew is used to install the `git` and `bash` utils, however `git` can be acquired directly from Apple through Xcode Command Line Tools, and `bash` comes packaged with the system. In addition, the tools are not currently in use by the hardening script.

Installing Brew in this way has the following security impacts:

- Brew is installed by directly retrieving the script from Github without additional verification.
- Brew itself is a vector for supply chain attacks as the software and repositories are community developed and maintained. In the scenario that one of these community maintained repositories or the software itself is compromised, this can present a security impact to the user of this script.

```
install_dependencies() {
    info "Installing dependencies..."

    # Check if Homebrew is installed
    if ! command -v brew &> /dev/null; then
        info "Installing Homebrew..."
        /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
    fi

    # Install necessary tools
    local packages=("bash" "git")
    for package in "${packages[@]}"; do
        if ! brew list "$package" &> /dev/null; then
            info "Installing $package..."
            brew install "$package"
        else
            success "$package is already installed"
        fi
    done
}
```

Recommendations

Do not install Brew if it is not absolutely required by the hardening scripts.

If Brew is required, consider packaging or referring to specific versions of the script that have been verified.

Resolution

The recommendation has been implemented in PR #20. The script no longer automatically installs the brew package manager and instead prompts the user to install out of band.

OPK-06	Cron Configuration Deleted Without Backup		
Asset	`linux/modules/services/cron.sh`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Low	Likelihood: Low

Description

The Linux hardening script deletes the `/etc/cron.deny` and `/etc/at.deny` files if they exist without first backing up the file.

While it is generally recommended to delete these files due to general deny list flaws and the invalid security posture it presents, it is possible for the system to continue to utilise them and the contents of the files should be verified and back ups created prior to deletion.

```
remove_deny_files() {
    log_info "Removing legacy cron and at deny files"
    log_info "Switching to allowlist-based access control for scheduled tasks"
    for f in /etc/cron.deny /etc/at.deny; do
        if [[ -f "$f" ]]; then
            if [[ "${HARDEN_DRY_RUN:-false}" == "true" ]]; then
                log_info "[dry-run] remove $f"
            else
                rm -f "$f" || true
            fi
        fi
    done
}
```

Script runs the `rm -f` command on the `cron.deny` and `at.deny` configuration files without first performing a back up.

Recommendations

Verify the contents of and back up the configuration files prior to deleting them.

Resolution

The recommendation has been implemented in PR #20. The script now performs a backup of the `cron.deny` and `at.deny` if they contain contents prior to deletion.

OPK-07	Fragile Update Configuration		
Asset	`linux/modules/system/updates.sh`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Low	Likelihood: Low

Description

The Linux hardening script runs the `upgrade`, `dist-upgrade` and `autoremove` operations which is considered an agressive update combination and has been known to cause system failure on distrubutions that utilise rolling updates, non-LTS distribution or scenarios where a major jump between versions has occurred.

This occurs due to how `dist-upgrade` will automatically remove packages to satisfy dependencies, where the regular upgrade option will not perform this action.

In addition, the script does not perform any check that the upgrade was succesful prior to logging "System update completed".

```
update_debian_system() {
    log_info "Updating Debian/Ubuntu system packages"

    if [[ "${HARDEN_DRY_RUN:-false}" == "true" ]]; then
        log_info "[dry-run] apt-get update && apt-get upgrade -y && apt-get dist-upgrade -y && apt-get autoremove -y"
        return
    fi

    log_info "Updating package cache..."
    apt-get update -y

    log_info "Upgrading installed packages..."
    DEBIAN_FRONTEND=noninteractive apt-get upgrade -y

    log_info "Performing distribution upgrade..."
    DEBIAN_FRONTEND=noninteractive apt-get dist-upgrade -y || true

    log_info "Removing unused packages..."
    apt-get autoremove -y || true

    log_info "System update completed"
}
```

Recommendations

Do not automatically perform the `dist-upgrade` operation.

Consider perfoming both commands with the `--simulate` option to prevent actual system changes and compare the two, and upon detecting differences between the two prompt the user to suggest the aggressive update option.

Perform a check to determine if the update was successful prior to logging "System updated completed".

Resolution

The recommendation has been implemented in PR #20. A check is now performed to determine the difference between simulations of `apt get upgrade` and `apt get dist-upgrade` and prompts the user prior to performing an aggressive upgrade.

OPK-08	Filesystem Remount Not Persistent		
Asset	`linux/modules/filesystem/permissions.sh`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Low	Likelihood: Low

Description

The `secure_temp_directories` function uses the `mount` command to remount directories with secure permissions configurations, however this method does not persist through a system restart.

Upon restart, these directories will be remounted with permissions as defined in the `/etc/fstab` file.

```
secure_temp_directories() {
    local targets=(/tmp /var/tmp)
    for t in "${targets[@]}"; do
        if mount | awk '{print $3}' | grep -qx "$t"; then
            if [[ "${HARDEN_DRY_RUN:-false}" == "true" ]]; then
                log_info "[dry-run] mount -o remount,nodev,nosuid,noexec $t"
            else
                mount -o remount,nodev,nosuid,noexec "$t" 2>/dev/null || true
            fi
        fi
    done
}
```

Recommendations

To permanently change directory permissions, entries in the `/etc/fstab` permissions table must be changed.

Resolution

The recommendation has been implemented in PR #20. The `secure_temp_directories` function has been altered to modify the `/etc/fstab` and permanently apply the secure permissions changes.

OPK-09	Root Account Shell Change Breaks Login		
Asset	`linux/modules/access/users.sh`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Low	Likelihood: Low

Description

The Linux hardening script sets the the login shell for all user accounts with id less than 1000 to `/usr/sbin/nologin`, preventing interactive use of these accounts, however this also includes the root account.

Changing the shell of root can have unintended consequences on system usage, for example this directly impacts running the `su` command and may cause issues with scripts running at high privilege.

```
lock_system_accounts() {
    log_info "Securing system accounts by restricting shell access"
    log_info "Identifying and locking non-essential system accounts"
    while IFS=: read -r name _ uid gid gecos home shell; do
        if [[ $uid -lt 1000 && $shell != */nologin && $shell != */false ]]; then
            if [[ "${HARDEN_DRY_RUN:-false}" == "true" ]]; then
                log_info "[dry-run] usermod -s /usr/sbin/nologin $name"
            else
                command -v usermod >/dev/null 2>&1 && usermod -s /usr/sbin/nologin "$name" 2>/dev/null || true
            fi
        fi
    done </etc/passwd
}
```

Recommendations

Explicitly exclude the root account from the `lock_system_accounts` function.

Resolution

The recommendation has been implemented in PR #20. The `lock_system_accounts` function no longer changes the login shell for the root account.

OPK-10	Flawed Firewall Configuration		
Asset	`linux/modules/networl/firewall.sh`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Medium	Likelihood: Low

Description

Multiple flaws have been identified in the Linux hardening firewall configuration that increase the attack surface of the system:

- The UFW configuration in the paranoid hardening profile configures a limit on port 22 which implicitly allows incoming connections on the port, however the expected behaviour for paranoid mode is that the SSH service is running on a randomised port and not the default port 22.

```
# Apply paranoid mode restrictions if enabled
if [[ "${HARDEN_PROFILE:-}" == "paranoid" ]]; then
    log_info "Applying paranoid mode restrictions"
    # Rate limit incoming connections
    ufw limit 22/tcp comment 'Rate limit SSH'
    # Log all blocked traffic
    ufw logging high
else
    ufw logging low
fi
```

- The `firewalld` configuration does not check that the SSH service is running or enabled prior to configuring an allow rule. On the paranoid profile, this configures an allow incoming rule on a random TCP port.

```
# Get SSH port from profile
local ssh_port
ssh_port=$(get_profile_setting "SSH_PORT")

if [[ "${HARDEN_DRY_RUN:-false}" == "true" ]]; then
    log_info "[dry-run] Configuring FirewallD with SSH on port $ssh_port"
    return
fi

log_info "Setting default zone to public"
firewall-cmd --set-default-zone=public || true

log_info "Configuring SSH access"
if [[ "$ssh_port" != "22" ]]; then
    log_info "Adding custom SSH port $ssh_port"
    firewall-cmd --permanent --add-port="$ssh_port/tcp" || true
else
    log_info "Adding standard SSH service"
    firewall-cmd --permanent --add-service=ssh || true
fi
```

- UFW has logic to detect if `apache`, `nginx` or `httpd` services are running, however does not check what port they are running on and whitelists TCP port 80 and 443 regardless of the service configurations.

```
# Check for HTTP/HTTPS (Apache or Nginx)
if systemctl is-active apache2.service >/dev/null 2>&1 || \
   systemctl is-active httpd.service >/dev/null 2>&1 || \
   systemctl is-active nginx.service >/dev/null 2>&1; then
    services+=("web")
fi

echo "${services[*]}"
```

Recommendations

The following recommendations must be implemented to resolve this issue:

- Ensure the paranoid profile is correctly setting a limit for the randomised port number and not port 22.
- Perform a check that the SSH service is running and enabled prior to allowing access on ports through either firewalld or UFW
- Do not allow incoming on arbitrary web services without explicit agreement of the user

Resolution

The recommendation has been implemented in PR #23. The following fixes have been implemented:

- The paranoid profile now explicitly denies incoming connections on TCP port 22.
- Checks are now in place to determine if the service was running prior to configuring an allow rule on its associated TCP port.
- Service detection logic now determines the port the service is running on prior to configuring an allow rule.

OPK-11	Flawed SSH Port Randomisation		
Asset	`linux/modules/core/common.sh`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Low	Likelihood: Low

Description

The port randomisation functionality defined within the Linux hardening script is flawed and may lock out a user from remote access to the system.

The SSHD (Secure Shell Daemon) listening port is set to a number within the private port range (TCP 49152 - 65535) without first checking that the port is not already in use. This range is commonly used by applications as an ephemeral port range, and if in use when executing the hardening script, may lock out the user from remote access.

```
init_paranoid_profile() {
    log_info "Initializing paranoid security profile"
    PROFILE_SETTINGS["PASSWORD_MAX_DAYS"]=30
    PROFILE_SETTINGS["PASSWORD_MIN_DAYS"]=21
    PROFILE_SETTINGS["PASSWORD_WARN_AGE"]=14
    PROFILE_SETTINGS["SSH_PORT"]="$(shuf -i 49152-65535 -n 1)"  # Random high port
    PROFILE_SETTINGS["SSH_PERMIT_ROOT_LOGIN"]="no"
    PROFILE_SETTINGS["SSH_PASSWORD_AUTH"]="no"
    PROFILE_SETTINGS["FIREWALL_DEFAULT_POLICY"]="drop"
    PROFILE_SETTINGS["IPV6_ENABLED"]="no"
    PROFILE_SETTINGS["AUDITD_ENABLED"]="yes"
    PROFILE_SETTINGS["FAIL2BAN_ENABLED"]="yes"
}
```

Recommendations

Upon selecting a random port, perform a check to determine if another service is already listening on the port.

Resolution

The recommendation has been implemented in PR #21. The port randomisation feature now performs a check to ensure that the chosen port is not already listening on an interface.

OPK-12	SSH Service Force Started		
Asset	`linux/modules/access/ssh.sh`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Medium	Likelihood: Low

Description

The Linux hardening script automatically starts the SSHD (Secure Shell Daemon) service regardless of the previous state of the service.

As this action may be executed without the knowledge of the user, in the scenario where the SSHD binaries are installed but the service is not running, the attack surface of the device is increased.

In the case that this is running with a network interface directly on the Internet, as the case may be on a cloud hosted compute service, this can expose the SSH service directly to the Internet upon running the hardening script.

```
restart_ssh_service() {
    log_info "Applying new SSH configuration by restarting the service"
    if [[ "${HARDEN_DRY_RUN:-false}" == "true" ]]; then
        log_info "[dry-run] systemctl reload sshd || service ssh restart"
    else
        systemctl reload sshd 2>/dev/null || systemctl restart ssh 2>/dev/null || true
    fi
    log_info "SSH service has been configured and restarted"
}
```

Recommendations

Prior to interacting with the SSHD service, perform a check to determine whether the service is already running. If the service is not already running, do not perform a service start or restart action.

Resolution

The recommendation has been implemented in PR #21. Checks are now performed to determine if the SSH service is already running prior to restarting.

OPK-13	Flawed Backup Behaviour On Linux Hardening Scripts		
Asset	`linux/modules/core/common.sh, linux/*.sh`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Low	Likelihood: Medium

Description

The backup function defined in the Linux hardening script overwrites the existing backup file whenever the function is called multiple times for the same file. When this function is called multiple times, the original backup of the file is overwritten.

This specific scenario occurs under `ssh.sh` where the backup function is transitively called 15 times, repeatedly overwriting the original backup of the file with new changes.

```
backup_file() {
  local file="$1"
  local stamp="$RUN_STAMP"
  local rel
  rel="${file#/}"
  local dest_dir="$HARDEN_BACKUP_DIR_BASE/$stamp/$(dirname "$rel")"

  log_verbose "Creating backup of file: $file"
  mkdir -p "$dest_dir"
  if [[ -f "$file" ]]; then
    cp -a "$file" "$dest_dir/" || true
    log_verbose "Backup created in: $dest_dir"
  else
    log_verbose "File $file does not exist - no backup needed"
  fi
}
```

The same behaviour was identified under the following functions:

- `kernel.sh` - `harden_ipv4_settings`, `harden_ipv6_settings`, `harden_kernel_settings`
- `users.sh` - `secure_login_defs`
- `backup.sh` - `configure_journald`
- `sudo.sh` - `configure_sudo_defaults`

Recommendations

Perform a check to determine if a backup file already exists. As the backup directory is created with a timestamp in the name, repeated runs of the hardening script will not conflict with eachother.

Due to this, a simple check if the backup file already exists will preserve the original file backup without significant alteration to the caller functions.

Resolution

The recommendation has been implemented in PR #21. The `backup_file` function now performs a check to determine if the backup file already exists to prevent the script from overwriting the original backup file.

OPK-14	Fragile Sudoers File Manipulation		
Asset	`linux/modules/access/sudo.sh`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Low	Likelihood: Low

Description

The Linux hardening script applies configuration changes directly to the `sudoers` file, this is considered unsafe as any flaw or system failure that results in a syntax error when the file is parsed can result in privileged system access being restricted for all users.

```
configure_sudo_defaults() {
    backup_file /etc/sudoers

    # Configure logging
    apply_line /etc/sudoers "^Defaults\s+logfile\s*=" "Defaults    logfile=/var/log/sudo.log"
    ensure_owner_perm /var/log/sudo.log root root 0600

    # Set security options
    apply_line /etc/sudoers "^Defaults\s+passwd_tries\s*=" "Defaults    passwd_tries=3"
    apply_line /etc/sudoers "^Defaults\s+requiretty\s*=" "Defaults    requiretty"
    apply_line /etc/sudoers "^Defaults\s+use_pty\s*=" "Defaults    use_pty"
}
```

Recommendations

A combination of two safer methods of editing the sudoers configuration are recommended:

- Copy the `sudoers` file to a temporary file, make the changes to the temporary file, perform a validation of the file and finally overwrite the `/etc/sudoers` file. Validation can be performed using the visudo command `visudo -cf /path/to/su`
- Utilise the `/etc/sudoers.d/` directory to apply hardening configurations, as these will override configurations defined in the `/etc/sudoers` file

Resolution

The recommendation has been implemented in PR #21. The `configure_sudo_defaults` function now performs a check to determine that the modified sudoers file is valid prior to installing it.

OPK-15	Inconsistently Defined Harden_Profile Configurations		
Asset	`linux/modules/core/common.sh`, `linux/modules/access/ssh.sh`, `linux/modules/network/fail2ban.sh`, `linux/modules/network/firewall.sh`, `linux/README.md`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Low	Likelihood: Low

Description

The Linux hardening scripts inconsistently define and apply hardening profile configurations.

Under `common.sh`, profiles are defined and set values in the array `PROFILE_SETTINGS` with the configurations associated with the relevant profile, however throughout the script rather than referring to the array, functions instead refer directly to the `HARDEN_PROFILE` variable.

In addition, further conflicting variables are set in `main.sh` depending on the profile setting rather than relying on the profile array.

In some circumstances, the array and hardcoded profile check configuration conflict with eachother and what is advised in the `README.md` file.

The following instances were identified:

- In `fail2ban.sh`, the script checks the `HARDEN_PROFILE` setting rather than referring to the `FAIL2BAN_ENABLED` key in the `PROFILE_SETTINGS` array. In addition, The `init_recommended_profile` function in `common.sh` indicates `fail2ban` should be enabled, however the `README.md` advises it will not be enabled which aligns with the hardcoded configuration.
- In `firewall.sh`, the script has a hardcoded check for `HARDEN_PROFILE` however then applies a hardcoded configuration for TCP port 22 rather than utilising the port defined in the profile through the randomisation feature.
- In `permissions.sh`, the script has a hardcoded check for `HARDEN_PROFILE` rather than relying on the `PROFILE_SETTINGS` array.
- In `ssh.sh`, the script has all configurations hardcoded rather than relying on the configuration associated with the profile. In addition, the script fails to set the randomised port as defined by `README.md` and the paranoid profile configuration.

```
init_recommended_profile() {
    log_info "Initializing recommended security profile"
    PROFILE_SETTINGS["PASSWORD_MAX_DAYS"]=90
    PROFILE_SETTINGS["PASSWORD_MIN_DAYS"]=7
    PROFILE_SETTINGS["PASSWORD_WARN_AGE"]=14
    PROFILE_SETTINGS["SSH_PORT"]="22"
    PROFILE_SETTINGS["SSH_PERMIT_ROOT_LOGIN"]="no"
    PROFILE_SETTINGS["SSH_PASSWORD_AUTH"]="no"
    PROFILE_SETTINGS["FIREWALL_DEFAULT_POLICY"]="drop"
    PROFILE_SETTINGS["IPV6_ENABLED"]="yes"
    PROFILE_SETTINGS["AUDITD_ENABLED"]="yes"
    PROFILE_SETTINGS["FAIL2BAN_ENABLED"]="yes"
}
```

`common.sh` sets an array for configurations associated with a profile.

```
configure_fail2ban() {
    # Check if we are in paranoid mode
    if [[ "${HARDEN_PROFILE:-}" != "paranoid" ]]; then
        log_info "Fail2ban configuration skipped - only available in paranoid mode"
        return 0
    fi

    log_info "Configuring Fail2ban security policies (paranoid mode)"
    log_info "Setting up strict intrusion detection rules and ban policies"
```

`fail2ban.sh` has a hardcoded check for the `HARDEN_PROFILE` setting rather than relying on configured profile array.

```
configure_ssh_security() {
    # First check if SSH is installed
    if ! is_ssh_installed; then
        log_info "SSH server not detected - skipping SSH hardening"
        return 0
    fi

    log_info "SSH server detected - proceeding with security hardening"
    log_info "Configuring SSH security settings for enhanced protection"
    log_info "Setting up SSH protocol and authentication restrictions"
    # Basic SSH protocol and authentication settings
    ensure_sshd_conf Protocol 2
    ensure_sshd_conf PermitRootLogin no
    ensure_sshd_conf PasswordAuthentication no
    ensure_sshd_conf PubkeyAuthentication yes
    ensure_sshd_conf PermitEmptyPasswords no

    # Authentication and session settings
    ensure_sshd_conf ChallengeResponseAuthentication no
    ensure_sshd_conf UsePAM yes
    ensure_sshd_conf MaxAuthTries 3
    ensure_sshd_conf MaxSessions 5
    ensure_sshd_conf LoginGraceTime 30

    # Connection and forwarding settings
    ensure_sshd_conf X11Forwarding no
    ensure_sshd_conf AllowTcpForwarding no
    ensure_sshd_conf AllowAgentForwarding no
    ensure_sshd_conf ClientAliveInterval 300
    ensure_sshd_conf ClientAliveCountMax 2
```

`ssh.sh` defines specific configurations rather than relying on profile variables. This results in the paranoid mode configuration not correctly setting the SSH port to the random private port number.

Recommendations

Rely on one mechanism to define and apply profile configurations. Currently there are three mechanisms for profiles:

- `PROFILE_SETTINGS` array as defined in `common.sh`
- Functions directly referring to the variable `HARDEN_PROFILE`
- Additional configuration-specific variables defined in `main.sh` depending on the selected profile

In addition, ensure `README.md` documentation correctly correlates to profiles defined in the hardening script.

Resolution

The recommendation has been implemented in PR #21. Application of the harden profile value has been standardised across the hardening scripts.

OPK-16	Insecure Operational Practice Recommended By Windows Hardening Script Guide
Asset	`windows/README.md`
Status	**Resolved:** See Resolution
Rating	Informational

Description

The Windows hardening script `README.md` file specifies as the first and second options for running the script is that the user should paste a command into an administrative console which retrieves the script to be executed directly from the Internet.

This is an unsafe operational security practice and users should be advised to review scripts retrieved from the Internet prior to executing them in an administrative context.

```
### Option 1 - Run directly from PowerShell
Open **PowerShell as Administrator** and paste this one-liner:

Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass -Force ; irm
      ↪ 'https://raw.githubusercontent.com/Opsek/OSs-security/refs/heads/main/windows/hardening-windows.ps1' | iex


### Option 2 - Run from CMD
Open **Command Prompt as Administrator** and paste this one-liner:

cmd
powershell -NoProfile -ExecutionPolicy Bypass -Command "irm
      ↪ 'https://raw.githubusercontent.com/Opsek/OSs-security/refs/heads/main/windows/hardening-windows.ps1' | iex"
```

Raw excerpt from the `README.md` file that instructs a user to execute the script in one-line format directly in an administrative context.

Recommendations

Advise the user to always review code prior to execution, and avoid instructing users to blindly retrieve and execute code in administrative contexts.

Resolution

The recommendation has been implemented in PR #21. The `README.md` file now advises the user never to directly execute scripts from the internet without prior analysis, and instructs the user to review the script prior to execution.

OPK-17	Windows Hardening Script Improvement Suggestions
Asset	`Windows Hardening Script`
Status	**Resolved:** See Resolution
Rating	Informational

Description

The following are additions and improvements recommended for the Windows hardening script, to be considered and added at the discretion of the maintainers:

- Disable OneDrive, this is a cloud storage mechanism packaged with Windows and should be disabled if not in use. This is best handled through group policy configuration.
- Perform a check if there are any online Microsoft accounts on the system, as opposed to local users. Microsoft accounts can utilise Internet-based authentication and present a direct backdoor into the system. This check can be performed in powershell using the command `Get-LocalUser | Select Name, PrincipalSource`
- Enforce password entry on return from screen saver/power saving states. This can be enforced through group policy configuration. Note, this configuration may warrant a less aggressive screen timeout than is currently implemented in the hardening script.
- Configure File Explorer to show the file extensions for known file types, preventing files from attempting to hide themselves through this configuration. For example `malicious-file.txt.exe` would show up as `malicious-file.txt` if this configuration is not enabled. This can be enabled through registry edit.
- Disable AI on Windows where possible. Windows is integrating AI features directly into both the operating system and its bundled applications, for example notepad and paint. These features present both a direct interface for Microsoft to capture sensitive information on a system, for example credentials in a configuration file or user web browsing behaviour, and also present utilities for a threat actor to interact with in a compromise scenario. Note that Microsoft brands multiple AI products as `copilot` and investigation into how to disable these individual features must be performed

Recommendations

Consider addition of the potential improvements identified to the Windows hardening script.

Resolution

The recommendation has been implemented in PR #27. The following improvements have been added to the Windows hardening scripts:

- Detections for Microsoft One Drive have been implemented.
- Detections for Microsoft online accounts have been implemented.
- A configuration to enforce password on wake and return from screensaver has been added.

- File extensions are now explicitly shown.
- Detection for Microsoft Co-Pilot has been implemented.

OPK-18	Suggested Alternative Hardening Mechanism For MacOS
Asset	`macOS Hardening Script`
Status	**Resolved:** See Resolution
Rating	Informational

Description

Apple natively supports hardening of devices through Mobile Device Mangement (MDM) software, and is a mechanism that should be considered for enforcement of hardening controls for at least one profile. This is recommended due to how Apple frequently deprecates configurations that are applied by hardening guidelines such as CIS Benchmark, and disabling of integrated features through direct manupulation of configuration files can result in further unexpected system impacts.

As of macOS 26 the MDM profile cannot be intalled via the command line, however the user can still install a custom local profile through the Device Management tab in the System Settings application. It is also trivial for a profile to be removed and changed, making this hardening action easy to reverse or change relevant configurations.

The following are policies that can be enforced through MDM, however this list is not exhaustive:

- Disable automatic login
- Set password policy
- Enforce filevault encryption
- Harden privacy preferences configurations (for example, screen recording, microphone permissions)
- Enforce gatekeeper to restrict execution to only signed applications/extensions
- Configure always-on
- Configure custom DNS settings
- Harden Safari and enforce content filtering
- Enforce automated security updates
- Configure system auditing and log retention
- Disable Applie Intelligence features
- Disable sharing features such as Air Drop
- Explicit disable of remote access mechanisms such as SSH.

Recommendations

Consider implementing a profile that generates a hardened MDM policy for the user to install through the System Settings application.

While not all configurations implemented by the current iteration of hardening scripts can be covered by MDM profiles, this can help provide a baseline security posture that most critically is supported by Apple, will be maintained through system upgrades and is trivial to later perform policy changes.

Resolution

The recommendation has been implemented in PR #27. Apple configuration profiles have now been added as an option for macOS hardening.

OPK-19	MacOS Hardening Script Specifies Support For Outdated Operating System
Asset	`macos/main.sh`
Status	**Resolved:** See Resolution
Rating	Informational

Description

The macOS hardening script explicitly states support for macOS Venture (13), however this operating system is at end of life and will no longer receive security updates from Apple.

It is recommended that the user be notified that the device no longer receives these updates when running the hardening script on the end of life operating systems.

```
# ==============================================================================
# macOS Security Hardening Script - OPSEK (Modular Version)
# ==============================================================================
# Version: 0.1.0
# Compatible with: macOS 13+ (Ventura, Sonoma, Sequoia)
#
# FEATURES:
#   - CIS Benchmark compliance
#   - NIST 800-53 controls
#   - OPSEK project recommendations
#   - Multi-profile hardening (basic, moderate, strict, paranoid)
#   - Automated backup and rollback capabilities
#   - Comprehensive logging and audit trail
#   - Modular architecture for collaborative development
# ==============================================================================
```

Excerpt from `main.sh` stating that macOS 13 is supported.

Recommendations

Notify the user that this operating system version is no longer receiving security updates from Apple when running the hardening script.

Resolution

The recommendation has been implemented in PR #23. The script now alerts the user that macOS 13 is no longer receiving security updates from Apple when that version of the operating system is detected.

OPK-20	Linux Hardening Script Improvement Opportunities
Asset	`Linux Hardening Scripts, linux/modules/core/common.sh, linux/modules/filesystem/permissions.sh`
Status	**Closed:** See Resolution
Rating	Informational

Description

The following are additions and improvements recommended for the Linux hardening script, to be considered and added at the discretion of the maintainers:

- Install the AIDE (Advanced Intrustion Detection Environment) utility, which can be retrieved from standard apt and RPM repositories across many distributions of Linux.

  AIDE is a mechanism for checking local file integrity, and can be set up as a scheduled task to repeatedly check for and report changes to critical system files.

- Consider installing USBGuard on the paranoid profile.

  USBGuard is an allow-listing mechanism for USB devices to mitigate scenario of a rogue device connecting to a system, and is available from standard apt and RPM repositories across multiple distributions of Linux.

  Initial configuration and profiling of system USB devices can be performed to build a baseline allowlist, however it is critical to advise the user that this control will be enabled as if it is misconfigured it may impact internal device functionality such as laptop keyboards, webcams and trackpads which are handled as USB devices on some machines. Devices such as Yubikeys would also be prevented from connecting to the system unless first authorised.

- Ensure popularity-contest package has been disabled on Debian/Ubuntu based distributions.

  Popularity-contest is a telemetry mechanism that reports system information back to the distribution maintainer.

- Configure unattended-upgrades for security updates on Debian/Ubuntu based distributions.

  Unattended-upgrades is a mechanism that allows `apt` to periodically query for updates, and is configured by default to automatically install only security updates.

- As part of the hardening script checks, perform a check to determine if secureboot is enabled and if a known filesystem encryption mechanism is implemented.

  Enabling these would be outside of the scope of an automated hardening script, however the user should be advised of the risks of having these controls disabled.

- Currently the `ensure_owner_perm` function in `common.sh` performs `chown` and `chmod` of the file regardless of the current setting, it is advised to not perform this action if the target file already meets the ownership and permissions specifications required.

- The same `ensure_owner_perm` function performs a backup of the file being modified prior to adjusting the permissions, however certain system files such as `/etc/shadow` that are passed to this function may not be appropriate to back up in this way as they contain password hashes of user accounts.

- The `secure_authentication_files` function in `permissions.sh` currently adjusts the permissions of sensitive authentication files, however does not check for the permissions of their associated backup files, `/etc/passwd-`, `/etc/group-`, `/etc/shadow-` and `/etc/gshadow-`

Recommendations

Consider addition of the potential improvements identified to the Linux hardening scripts.

Resolution

The Opsek team will consider the implementation of the improvement suggestions at a later date. As this finding is informational and does not present a security impact, it is now closed.

OPK-21	Aggressive Disable Of System Services
Asset	`linux/modules/services/services.sh`
Status	**Resolved:** See Resolution
Rating	Informational

Description

The Linux hardening scripts disables system services that may be too aggressive for a recommended system profile:

- `avahi-daemon` is the service responsible for multicast DNS operations, disabling this service will generally impact local network discovery operations that include `.local` hostname discovery; printer (IPP) and airprint functions, "casting" functionality and general IOT device integrations.
- `cups` is responsible for handling print operations on the local system, disabling of this service may have an impact to any print operation occuring on the local system.

```
disable_unnecessary_services() {
    log_info "Identifying and disabling unnecessary network services"
    local disable=(avahi-daemon cups telnet vsftpd xinetd tftp)
    log_info "Services to be disabled: ${disable[*]}"
    for s in "${disable[@]}"; do
        if systemctl list-unit-files | grep -q "^$s\.service"; then
            if [[ "${HARDEN_DRY_RUN:-false}" == "true" ]]; then
                log_info "[dry-run] systemctl disable --now $s"
            else
                systemctl disable --now "$s" 2>/dev/null || true
            fi
        fi
    done
}
```

Hardening script disables `avahi-daemon` and `cups` services automatically.

Recommendations

Consider only disabling the `avahi-daemon` and `cups` services under the paranoid profile, or alternatively alerting the user to what functionalities may be impacted through running the hardening script.

Resolution

The recommendation has been implemented in PR #23. The user is now prompted to confirm if `cups` and `avahi-daemon` services should be disabled, and discloses the potential functional system impacts that this may cause.

OPK-22	Inconsistent Dry Run Behaviour
Asset	`linux/modules/core/common.sh`
Status	**Resolved:** See Resolution
Rating	Informational

Description

The Linux hardening script continues to perform the following filesystem operations (outside of logging behaviour) under a dry run:

- Backup operations under the `sysctl_apply`, `sed_comment_out`, `apply_line`, `ensure_owner_perm` functions all continue to create backup files
- The script creates the directory `/etc/ssh` under the `ensure_sshd_conf` function if the directory does not already exist

```
backup_file() {
  local file="$1"
  local stamp="$RUN_STAMP"
  local rel
  rel="${file#/}"
  local dest_dir="$HARDEN_BACKUP_DIR_BASE/$stamp/$(dirname "$rel")"

  log_verbose "Creating backup of file: $file"
  mkdir -p "$dest_dir"
  if [[ -f "$file" ]]; then
    cp -a "$file" "$dest_dir/" || true
    log_verbose "Backup created in: $dest_dir"
  else
    log_verbose "File $file does not exist - no backup needed"
  fi
}
```

The `backup_file` function does not contain a check for dry run. This function is directly and transitively called throughout the script.

```
ensure_sshd_conf() {
  local key="$1" value="$2"
  local file="/etc/ssh/sshd_config"
  log_info "Setting SSH configuration: $key = $value"
  mkdir -p /etc/ssh
  apply_line "$file" "${key//\//\/}" "$key $value"
}
```

The `ensure_sshd_conf` function creates the `/etc/ssh` directory regardless of whether the script is in a dry run.

Recommendations

Perform a check that the script is runing in dry run mode before all filesystem write operations unrelated to logging actions.

Resolution

The recommendation has been implemented in PR #23. The scripts have been modified to prevent write and backup operations when executing a dry run operation.

OPK-23	Sudo Configuration Improvement Opportunities
Asset	`linux/modules/access/sudo.sh`
Status	**Closed:** See Resolution
Rating	Informational

Description

Further improvements to the sudo hardening configuration have been identified:

- Check for instances of `NOPASSWD` in file. This configuration allows an account to perform the sudo command without verifying the account password and is a common vector for privilege escalation
- Check that the `secure_path` default is set. This mitigates exploitation of sudo configurations that do not set the absolute path for commands
- Perform hardening checks on files in `/etc/sudoers.d/` directory. Files in this directory are also considered as sudo configuration files and take priority over the `/etc/sudoers` file should they have conflicting entries.

Recommendations

Consider implementation of the hardening configurations identified:

- Check for instances of `NOPASSWD`
- Validate that `secure_path` default is set
- Applying current hardening checks to files in `/etc/sudoers.d/` directory

Resolution

The Opsek team will consider the implementation of the improvement suggestions at a later date. As this finding is informational and does not present a security impact, it is now closed.

Appendix A Vulnerability Severity Classification

This security review classifies vulnerabilities based on their potential impact and likelihood of occurance. The total severity of a vulnerability is derived from these two metrics based on the following matrix.

	High	Medium	High	Critical
Impact	Medium	Low	Medium	High
	Low	Low	Low	Medium
		Low	Medium	High
			Likelihood	

Table 1: Severity Matrix - How the severity of a vulnerability is given based on the *impact* and the *likelihood* of a vulnerability.