

# FileWave Automation Scripts

This article contains the scripts used by the Downloads page. The scripts are here for documentation purposes.

## Scripts

### FileWave Server Upgrade

[fwxserver\\_upgrade.sh](#) - This script is used by the Download page for FileWave Server upgrades on Debian. Some details;

```
# To run this script, use the following 1-liner:
# curl -fsSL https://kb.filewave.com/attachments/411 | sudo bash -s -- -v <version> -r <revision> [-b for beta] -y
# Example for version 15.5.0 with revision 1 in production:
# curl -fsSL https://kb.filewave.com/attachments/411 | sudo bash -s -- -v 15.5.0 -r 1 -p -y
```

#### ▼ fwxserver\_upgrade.sh

```
#!/bin/bash
# Documentation
# To run this script, use the following 1-liner:
# curl -fsSL https://kb.filewave.com/attachments/411 | sudo bash -s -- -v <version> -r <revision> [-b for beta] -y
# Example for version 15.5.0 with revision 1 in production:
# curl -fsSL https://kb.filewave.com/attachments/411 | sudo bash -s -- -v 15.5.0 -r 1 -p -y

# Ensure script is run with sudo/root privileges
if [ "$EUID" -ne 0 ]; then
    die "This script must be run with sudo or as root."
fi

# Ensure script is running on a Debian-based system
if ! grep -iq "debian" /etc/os-release; then
    die "This script must be run on a Debian-based system."
fi

# Set default values for version, revision, and server type
VERSION="15.5.0"
REVISION="1"
SERVER_TYPE="prod"
BASE_URL="https://fwdl.filewave.com"
auto_yes=false

if [ "$#" -eq 0 ]; then
    echo "Usage: curl -fsSL https://kb.filewave.com/attachments/409 | sudo bash -s -- -v <version> -r <revision> [-b for beta | -p for prod] [-y for auto-yes]"
    echo "
Options:"
    echo "  -v, --version      Specify the version of FileWave Server to install (e.g., 15.5.0)"
    echo "  -r, --revision     Specify the revision number (e.g., 1)"
    echo "  -b, --beta         Use beta server for downloads (default is production)"
    echo "  -p, --prod         Use production server for downloads"
    echo "  -y, --yes          Automatically answer 'yes' to all prompts"
    exit 1
fi

# Function to handle errors and display messages
die() {
    echo "[ERROR] $1" >&2
    exit 1
}

# Log all actions to syslog for audit purposes
LOG_FILE="/var/log/filewave_server_update.log"
touch "$LOG_FILE" || die "Failed to create log file $LOG_FILE. Check permissions."
chmod 644 "$LOG_FILE" || die "Failed to set permissions on log file $LOG_FILE."
if [[ ! -w "$LOG_FILE" ]]; then
```

```

        die "Cannot write to log file $LOG_FILE. Please check permissions."
    fi
    exec >>(stdbuf -oL tee -a "$LOG_FILE") 2>&1
    echo "Logging to $LOG_FILE"

# Parse input arguments
while [[ "$#" -gt 0 ]]; do
    case $1 in
        -v|--version)
            VERSION="$2"
            shift 2
            ;;
        -r|--revision)
            REVISION="$2"
            shift 2
            ;;
        -b|--beta)
            SERVER_TYPE="beta"
            shift
            ;;
        -p|--prod)
            SERVER_TYPE="prod"
            shift
            ;;
        -y|--yes)
            auto_yes=true
            shift
            ;;
        *)
            echo "Unknown option: $1"
            exit 1
            ;;
    esac
done

# Confirm actions with the user
echo "This script will update your OS and install/upgrade the FileWave Server packages."
echo "Version: $VERSION, Revision: $REVISION, Server Type: $SERVER_TYPE"
if [[ "$auto_yes" == "true" ]]; then
    confirm="yes"
else
    read -p "Do you wish to proceed? (yes/no): " confirm < /dev/tty
fi
if [[ ! "$confirm" =~ ^[Yy]([Ee][Ss])?$ ]]; then
    echo "Aborting installation as requested."
    exit 0
fi

# Set the appropriate URL based on server type
if [ "$SERVER_TYPE" == "beta" ]; then
    BASE_URL="https://fwbetas.filewave.com"
fi

# 1. Install Essential Tools
echo "Installing essential tools..."
apt-get clean || die "Failed to clean apt cache."
apt-get update -y || die "Failed to update package list."
apt-get --fix-broken install -y || die "Failed to fix broken installs from apt."
apt-get autoremove -y || die "Failed to autoremove from apt."

for dep in "curl" "zip" "gdebi"; do
    if ! command -v $dep &> /dev/null; then
        echo "$dep not found. Installing $dep..."
        apt-get install -y $dep || die "Failed to install $dep."
    fi
done

if ! dpkg-query -W -f='${Status}' net-tools 2>/dev/null | grep -q "install ok installed"; then
    echo "net-tools not found. Installing net-tools..."
    apt-get install -y net-tools || die "Failed to install net-tools."

```

```

fi

# Check the current version of FileWave Server
INSTALLED_VERSION=$(dpkg-query -W -f='${Version}' fwxsrv 2>/dev/null || true)
# If no FileWave Server package is installed, set the version to 1.0.0 for comparison purposes
if [[ -z "$INSTALLED_VERSION" || "$INSTALLED_VERSION" == "none" ]]; then
    echo "No FileWave Server packages are installed. Proceeding with initial installation of version $VERSION..."
    INSTALLED_VERSION="1.0.0"
fi

install_packages() {
    echo "Downloading and installing FileWave Server package..."
    PACKAGE="fwxsrv_${VERSION}_amd64.deb"
    DOWNLOAD_DIR="/tmp/filewave_install_$$"
    mkdir -p "$DOWNLOAD_DIR" || die "Failed to create download directory."
    trap 'rm -rf "$DOWNLOAD_DIR"' EXIT
    echo "Downloading $PACKAGE..."
    for i in {1..3}; do
        curl -fSL "$BASE_URL/$VERSION/$PACKAGE" -o "$DOWNLOAD_DIR/$PACKAGE" && break
        echo "Retrying download ($i/3)..."
        sleep 5
    done
    if [[ ! -f "$DOWNLOAD_DIR/$PACKAGE" ]]; then
        die "Failed to download $PACKAGE after multiple attempts."
    fi
    echo "Installing $PACKAGE..."
    gdebi -n "$DOWNLOAD_DIR/$PACKAGE" || die "Failed to install $PACKAGE"

    # Clean up the download directory
    if [[ -d "$DOWNLOAD_DIR" ]]; then
        rm -rf "$DOWNLOAD_DIR" || echo "Warning: Failed to remove download directory $DOWNLOAD_DIR"
    fi
}

# This is the meat of the script.
# The actions to take based on version.
if dpkg --compare-versions "$INSTALLED_VERSION" lt "$VERSION"; then
    echo "Upgrading to FileWave Server version $VERSION..."
    install_packages
elif dpkg --compare-versions "$INSTALLED_VERSION" eq "$VERSION"; then
    echo "FileWave Server is already at version $VERSION, no further installation required."
else
    echo "FileWave Server is newer than $VERSION and downgrade is not possible with this script."
fi

# Clean up the download directory
if [[ -d "$DOWNLOAD_DIR" ]]; then
    rm -rf "$DOWNLOAD_DIR" || echo "Warning: Failed to remove download directory $DOWNLOAD_DIR"
fi

# 4. OS Upgrade
echo "Warning: This script will upgrade the entire OS. This is required for security."
if [[ "$auto_yes" == "true" ]]; then
    upgrade_confirm="yes"
else
    read -p "Do you wish to proceed with the OS upgrade? (yes/no): " upgrade_confirm < /dev/tty
fi
if [[ ! "$upgrade_confirm" =~ ^[Yy]([Ee][Ss])?$ ]]; then
    echo "Skipping OS upgrade."
else
    echo "Upgrading the system..."
    apt-get update -y || die "Failed to update package list."
    DEBIAN_FRONTEND=noninteractive apt-get upgrade -y -o Dpkg::Options::="--force-confold" || die "System upgrade failed."
fi

# 5. Reboot the system to complete the installation
echo "The system will reboot in 15 seconds to complete the installation."
if [[ "$auto_yes" == "true" ]]; then

```

```

        reboot_confirm="yes"
    else
        read -p "Do you want to reboot now? (yes/no): " reboot_confirm < /dev/tty
    fi
    if [[ "$reboot_confirm" =~ ^[Yy]([Ee][Ss])?$ ]]; then
        echo "Rebooting system in 15 seconds... Please remember that you can check the log file at $LOG_FILE after
the reboot to see the full details of the update."
        sleep 15
        reboot
    else
        echo "Skipping reboot. Please remember to reboot manually to apply all changes."
    fi

# The end

```

## FileWave Booster Upgrade

[fwbooster\\_upgrade.sh](#) - This script is used by the Download page fro FileWave Booster upgrades on Debian. Some details;

```

# To run this script, use the following 1-liner:
# curl -fsSL https://kb.filewave.com/attachments/412 | sudo bash -s -- -v <version> -r <revision> [-b for beta] -y
# Example for version 15.5.0 with revision 1 in production:
# curl -fsSL https://kb.filewave.com/attachments/412 | sudo bash -s -- -v 15.5.0 -r 1 -p -y

```

### ▼ fwbooster\_upgrade.sh

```

#!/bin/bash
# Documentation
# To run this script, use the following 1-liner:
# curl -fsSL https://kb.filewave.com/attachments/412 | sudo bash -s -- -v <version> -r <revision> [-b for
beta] -y
# Example for version 15.5.0 with revision 1 in production:
# curl -fsSL https://kb.filewave.com/attachments/412 | sudo bash -s -- -v 15.5.0 -r 1 -p -y

# Ensure script is run with sudo/root privileges
if [ "$EUID" -ne 0 ]; then
    die "This script must be run with sudo or as root."
fi

# Ensure script is running on a Debian-based system
if ! grep -iq "debian" /etc/os-release; then
    die "This script must be run on a Debian-based system."
fi

# Set default values for version, revision, and server type
VERSION="15.5.0"
REVISION="1"
SERVER_TYPE="prod"
BASE_URL="https://fwdl.filewave.com"
auto_yes=false

if [ "$#" -eq 0 ]; then
    echo "Usage: curl -fsSL https://kb.filewave.com/attachments/409 | sudo bash -s -- -v <version> -r
<revision> [-b for beta | -p for prod] [-y for auto-yes]"
    echo "\nOptions:"
    echo "  -v, --version      Specify the version of FileWave Booster to install (e.g., 15.5.0)"
    echo "  -r, --revision     Specify the revision number (e.g., 1)"
    echo "  -b, --beta         Use beta server for downloads (default is production)"
    echo "  -p, --prod         Use production server for downloads"
    echo "  -y, --yes          Automatically answer 'yes' to all prompts"
    exit 1
fi

# Function to handle errors and display messages
die() {

```

```

    echo "[ERROR] $1" >&2
    exit 1
}

# Log all actions to syslog for audit purposes
LOG_FILE="/var/log/filewave_booster_update.log"
touch "$LOG_FILE" || die "Failed to create log file $LOG_FILE. Check permissions."
chmod 644 "$LOG_FILE" || die "Failed to set permissions on log file $LOG_FILE."
if [[ ! -w "$LOG_FILE" ]]; then
    die "Cannot write to log file $LOG_FILE. Please check permissions."
fi
exec >>(stdbuf -oL tee -a "$LOG_FILE") 2>&1
echo "Logging to $LOG_FILE"

# Parse input arguments
while [[ "$#" -gt 0 ]]; do
    case $1 in
        -v|--version)
            VERSION="$2"
            shift 2
            ;;
        -r|--revision)
            REVISION="$2"
            shift 2
            ;;
        -b|--beta)
            SERVER_TYPE="beta"
            shift
            ;;
        -p|--prod)
            SERVER_TYPE="prod"
            shift
            ;;
        -y|--yes)
            auto_yes=true
            shift
            ;;
        *)
            echo "Unknown option: $1"
            exit 1
            ;;
    esac
done

# Confirm actions with the user
echo "This script will update your OS and install/upgrade the FileWave Booster packages."
echo "Version: $VERSION, Revision: $REVISION, Server Type: $SERVER_TYPE"
if [[ "$auto_yes" == "true" ]]; then
    confirm="yes"
else
    read -p "Do you wish to proceed? (yes/no): " confirm < /dev/tty
fi
if [[ ! "$confirm" =~ ^[Yy]([Ee][Ss])?$ ]]; then
    echo "Aborting installation as requested."
    exit 0
fi

# Set the appropriate URL based on server type
if [ "$SERVER_TYPE" == "beta" ]; then
    BASE_URL="https://fwbetas.filewave.com"
fi

# 1. Install Essential Tools
echo "Installing essential tools..."
apt-get clean || die "Failed to clean apt cache."
apt-get update -y || die "Failed to update package list."
apt-get --fix-broken install -y || die "Failed to fix broken installs from apt."
apt-get autoremove -y || die "Failed to autoremove from apt."

for dep in "curl" "zip" "gdebi"; do

```

```

        if ! command -v $dep &> /dev/null; then
            echo "$dep not found. Installing $dep..."
            apt-get install -y $dep || die "Failed to install $dep."
        fi
    done

    if ! dpkg-query -W -f='${Status}' net-tools 2>/dev/null | grep -q "install ok installed"; then
        echo "net-tools not found. Installing net-tools..."
        apt-get install -y net-tools || die "Failed to install net-tools."
    fi

    # Check the current version of FileWave Booster
    INSTALLED_VERSION=$(dpkg-query -W -f='${Version}' fwbooster 2>/dev/null || true)
    # If no FileWave Booster package is installed, set the version to 1.0.0 for comparison purposes
    if [[ -z "$INSTALLED_VERSION" || "$INSTALLED_VERSION" == "none" ]]; then
        echo "No FileWave Booster packages are installed. Proceeding with initial installation of version $VERSION..."
        INSTALLED_VERSION="1.0.0"
    fi

    install_packages() {
        echo "Downloading and installing FileWave Booster package..."
        PACKAGE="fwbooster_${VERSION}_amd64.deb"
        DOWNLOAD_DIR="/tmp/filewave_install_$$"
        mkdir -p "$DOWNLOAD_DIR" || die "Failed to create download directory."
        trap 'rm -rf "$DOWNLOAD_DIR"' EXIT
        echo "Downloading $PACKAGE..."
        for i in {1..3}; do
            curl -fSL "$BASE_URL/$VERSION/$PACKAGE" -o "$DOWNLOAD_DIR/$PACKAGE" && break
            echo "Retrying download ($i/3)..."
            sleep 5
        done
        if [[ ! -f "$DOWNLOAD_DIR/$PACKAGE" ]]; then
            die "Failed to download $PACKAGE after multiple attempts."
        fi
        echo "Installing $PACKAGE..."
        gdebi -n "$DOWNLOAD_DIR/$PACKAGE" || die "Failed to install $PACKAGE"

        # Clean up the download directory
        if [[ -d "$DOWNLOAD_DIR" ]]; then
            rm -rf "$DOWNLOAD_DIR" || echo "Warning: Failed to remove download directory $DOWNLOAD_DIR"
        fi
    }

    # This is the meat of the script.
    # The actions to take based on version.
    if dpkg --compare-versions "$INSTALLED_VERSION" lt "$VERSION"; then
        echo "Upgrading to FileWave Booster version $VERSION..."
        install_packages
    elif dpkg --compare-versions "$INSTALLED_VERSION" eq "$VERSION"; then
        echo "FileWave Booster is already at version $VERSION, no further installation required."
    else
        echo "FileWave Booster is newer than $VERSION and downgrade is not possible with this script."
    fi

    # Clean up the download directory
    if [[ -d "$DOWNLOAD_DIR" ]]; then
        rm -rf "$DOWNLOAD_DIR" || echo "Warning: Failed to remove download directory $DOWNLOAD_DIR"
    fi

    # 4. OS Upgrade
    echo "Warning: This script will upgrade the entire OS. This is required for security."
    if [[ "$auto_yes" == "true" ]]; then
        upgrade_confirm="yes"
    else
        read -p "Do you wish to proceed with the OS upgrade? (yes/no): " upgrade_confirm < /dev/tty
    fi
    if [[ ! "$upgrade_confirm" =~ ^[Yy]([Ee][Ss])?$ ]]; then
        echo "Skipping OS upgrade."
    else

```

```

    echo "Upgrading the system..."
    apt-get update -y || die "Failed to update package list."
    DEBIAN_FRONTEND=noninteractive apt-get upgrade -y -o Dpkg::Options::="--force-confold" || die "System
upgrade failed."
fi

# 5. Reboot the system to complete the installation
echo "The system will reboot in 15 seconds to complete the installation."
if [[ "$auto_yes" == "true" ]]; then
    reboot_confirm="yes"
else
    read -p "Do you want to reboot now? (yes/no): " reboot_confirm < /dev/tty
fi
if [[ "$reboot_confirm" =~ ^[Yy]([Ee][Ss])?$ ]]; then
    echo "Rebooting system in 15 seconds... Please remember that you can check the log file at $LOG_FILE after
the reboot to see the full details of the update."
    sleep 15
    reboot
else
    echo "Skipping reboot. Please remember to reboot manually to apply all changes."
fi

# The end

```

## FileWave IVS Upgrade

[ivs\\_upgrade.sh](#) - This script is used by the Download page for IVS upgrades on Debian. Some details;

# To run this script, use the following 1-liner:  
# curl -fsSL https://kb.filewave.com/attachments/408 | sudo bash -s -- -v <version> -r <revision> [-b for beta] -y  
# Example for version 15.5.0 with revision 1 in production:  
# curl -fsSL https://kb.filewave.com/attachments/408 | sudo bash -s -- -v 15.5.0 -r 1 -p -y

### ▼ ivs\_upgrade.sh

```

#!/bin/bash
# Documentation
# To run this script, use the following 1-liner:
# curl -fsSL https://kb.filewave.com/attachments/408 | sudo bash -s -- -v <version> -r <revision> [-b for
beta] -y
# Example for version 15.5.0 with revision 1 in production:
# curl -fsSL https://kb.filewave.com/attachments/408 | sudo bash -s -- -v 15.5.0 -r 1 -p -y

# Ensure script is run with sudo/root privileges
if [ "$EUID" -ne 0 ]; then
    die "This script must be run with sudo or as root."
fi

# Ensure script is running on a Debian-based system
if ! grep -iq "debian" /etc/os-release; then
    die "This script must be run on a Debian-based system."
fi

# Set default values for version, revision, and server type
VERSION="15.5.0"
REVISION="1"
SERVER_TYPE="prod"
BASE_URL="https://fwdl.filewave.com"
auto_yes=false
setup_cron_job=false
install_packages=false

if [ "$#" -eq 0 ]; then
    echo "Usage: curl -fsSL https://kb.filewave.com/attachments/401 | sudo bash -s -- -v <version> -r
<revision> [-b for beta] [-p for prod] [-y for auto-yes]"

```

```

    echo "
Options:"
    echo "  -v, --version      Specify the version of FileWave IVS to install (e.g., 15.5.0)"
    echo "  -r, --revision    Specify the revision number (e.g., 1)"
    echo "  -b, --beta        Use beta server for downloads (default is production)"
    echo "  -p, --prod        Use production server for downloads"
    echo "  -y, --yes         Automatically answer 'yes' to all prompts"
    exit 1
fi

# Function to handle errors and display messages
die() {
    echo "[ERROR] $1" >&2
    exit 1
}

# Log all actions to syslog for audit purposes
LOG_FILE="/var/log/filewave_ivs_update.log"
touch "$LOG_FILE" || die "Failed to create log file $LOG_FILE. Check permissions."
chmod 644 "$LOG_FILE" || die "Failed to set permissions on log file $LOG_FILE."
if [[ ! -w "$LOG_FILE" ]]; then
    die "Cannot write to log file $LOG_FILE. Please check permissions."
fi

exec >>(stdbuf -oL tee -a "$LOG_FILE") 2>&1
echo "Logging to $LOG_FILE"

# Parse input arguments
while [[ "$#" -gt 0 ]]; do
    case $1 in
        -v|--version)
            VERSION="$2"
            shift 2
            ;;
        -r|--revision)
            REVISION="$2"
            shift 2
            ;;
        -b|--beta)
            SERVER_TYPE="beta"
            shift
            ;;
        -p|--prod)
            SERVER_TYPE="prod"
            shift
            ;;
        -y|--yes)
            auto_yes=true
            shift
            ;;
        *)
            echo "Unknown option: $1"
            exit 1
            ;;
    esac
done

# Confirm actions with the user
echo "This script will update your OS and install/upgrade the FileWave IVS packages."
echo "Version: $VERSION, Revision: $REVISION, Server Type: $SERVER_TYPE"
if [[ "$auto_yes" == "true" ]]; then
    confirm="yes"
else
    read -p "Do you wish to proceed? (yes/no): " confirm < /dev/tty
fi

if [[ ! "$confirm" =~ ^[Yy]([Ee][Ss])?$ ]]; then
    echo "Aborting installation as requested."
    exit 0
fi

# Set the appropriate URL based on server type

```



```

if [ "$SERVER_TYPE" == "beta" ]; then
    BASE_URL="https://fwbetas.filewave.com"
fi

# 1. Install Essential Tools
echo "Installing essential tools..."
apt-get clean || die "Failed to clean apt cache."
apt-get update -y || die "Failed to update package list."
apt-get --fix-broken install -y || die "Failed to fix broken installs from apt."
apt-get autoremove -y || die "Failed to autoremove from apt."

for dep in "python3" "curl" "zip" "gdebi"; do
    if ! command -v $dep &> /dev/null; then
        echo "$dep not found. Installing $dep..."
        apt-get install -y $dep || die "Failed to install $dep."
    fi
done

if ! dpkg-query -W -f='${Status}' net-tools 2>/dev/null | grep -q "install ok installed"; then
    echo "net-tools not found. Installing net-tools..."
    apt-get install -y net-tools || die "Failed to install net-tools."
fi

# Check current version
INSTALLED_VERSION=$(dpkg-query -W -f='${Version}' ivs-kernel 2>/dev/null || true)
if [[ -z "$INSTALLED_VERSION" || "$INSTALLED_VERSION" == "none" ]]; then
    die "No FileWave IVS packages are installed. Please use the FileWave IVS Appliance image to set up IVS initially."
fi

# Extract installed main version and revision
INSTALLED_MAIN_VERSION=$(echo "$INSTALLED_VERSION" | cut -d'-' -f1)
INSTALLED_REVISION=$(echo "$INSTALLED_VERSION" | cut -d'-' -f2)

if [[ -z "$INSTALLED_REVISION" ]]; then
    INSTALLED_REVISION=0
fi

# Prevent installing 15.5.0
if dpkg --compare-versions "$VERSION" eq "15.5.0"; then
    echo "Installing FileWave IVS 15.5.0 is not supported. Please install version 15.5.1 or newer."
    exit 1
fi

# Prevent downgrades or re-installing the same version
if dpkg --compare-versions "$INSTALLED_MAIN_VERSION" gt "$VERSION" || \
    ( dpkg --compare-versions "$INSTALLED_MAIN_VERSION" eq "$VERSION" && [[ "$INSTALLED_REVISION" -ge
"$REVISION" ] ] ); then
    echo "Installed version ($INSTALLED_VERSION) is newer or equal to the requested version ($VERSION-
$REVISION)."
    echo "Aborting to prevent downgrade or re-installation of the same version."
    exit 1
fi

install_packages() {
    echo "Downloading and installing FileWave IVS packages..."
    PACKAGE_ORDER=(
        "filewave-admin_${VERSION}_amd64.deb"
        "filewave-imaging-client_${VERSION}_amd64.deb"
        "ivs-kernel-${VERSION}-${REVISION}.x86_64.deb"
        "filewave-ivs_${VERSION}_amd64.deb"
    )
    DOWNLOAD_DIR="/tmp/filewave_install_$$"
    mkdir -p "$DOWNLOAD_DIR" || die "Failed to create download directory."
    trap 'rm -rf "$DOWNLOAD_DIR"' EXIT
    for package in "${PACKAGE_ORDER[@]"; do
        echo "Downloading $package..."
        for i in {1..3}; do
            curl -fSL "$BASE_URL/$VERSION/$package" -o "$DOWNLOAD_DIR/$package" && break
            echo "Retrying download ($i/3)..."
        done
    done
}

```

```

        sleep 5
    done
    if [[ ! -f "$DOWNLOAD_DIR/$package" ]]; then
        die "Failed to download $package after multiple attempts."
    fi
    echo "Installing $package..."
    gdebi -n "$DOWNLOAD_DIR/$package" || die "Failed to install $package"
done

# Clean up the download directory
if [[ -d "$DOWNLOAD_DIR" ]]; then
    rm -rf "$DOWNLOAD_DIR" || echo "Warning: Failed to remove download directory $DOWNLOAD_DIR"
fi
}

# If we reach this point, the requested version is newer than the installed version.
echo "Upgrading from $INSTALLED_VERSION to $VERSION-$REVISION..."
install_packages

# Clean up the download directory
if [[ -d "$DOWNLOAD_DIR" ]]; then
    rm -rf "$DOWNLOAD_DIR" || echo "Warning: Failed to remove download directory $DOWNLOAD_DIR"
fi

# 4. OS Upgrade
echo "Warning: This script will upgrade the entire OS. This is required for security."
if [[ "$auto_yes" == "true" ]]; then
    upgrade_confirm="yes"
else
    read -p "Do you wish to proceed with the OS upgrade? (yes/no): " upgrade_confirm < /dev/tty
fi
if [[ ! "$upgrade_confirm" =~ ^[Yy]([Ee][Ss])?$ ]]; then
    echo "Skipping OS upgrade."
else
    echo "Upgrading the system..."
    apt-get update -y || die "Failed to update package list."
    DEBIAN_FRONTEND=noninteractive apt-get upgrade -y -o Dpkg::Options::="--force-confold" || die "System
upgrade failed."
fi

# 5. Reboot the system to complete the installation
echo "The system will reboot in 15 seconds to complete the installation."
if [[ "$auto_yes" == "true" ]]; then
    reboot_confirm="yes"
else
    read -p "Do you want to reboot now? (yes/no): " reboot_confirm < /dev/tty
fi
if [[ "$reboot_confirm" =~ ^[Yy]([Ee][Ss])?$ ]]; then
    echo "Rebooting system in 15 seconds... Please remember that you can check the log file at $LOG_FILE after
the reboot to see the full details of the update."
    sleep 15
    reboot
else
    echo "Skipping reboot. Please remember to reboot manually to apply all changes."
fi

# The end

```