scripts en manuele commando's:

script 1:

```
#!/usr/bin/env bash
################################################################################
#
# Bash Remediation Script for Standard System Security Profile for Ubuntu 22.04
#
# Profile Description:
# This profile contains rules to ensure standard security baseline of an Ubuntu 22.04 system. Regardless of your system's workload all of these checks should pass.
#
# Profile ID:  xccdf_org.ssgproject.content_profile_standard
# Benchmark ID:  xccdf_org.ssgproject.content_benchmark_UBUNTU_22-04
# Benchmark Version:  0.1.67
# XCCDF Version:  1.2
#
# This file was generated by OpenSCAP 1.3.7 using:
# $ oscap xccdf generate fix --profile xccdf_org.ssgproject.content_profile_standard --fix-type bash xccdf-file.xml
#
# This Bash Remediation Script is generated from an OpenSCAP profile without preliminary evaluation.
# It attempts to fix every selected rule, even if the system is already compliant.
#
# How to apply this Bash Remediation Script:
```

```
# $ sudo ./remediation-script.sh
#
################################################################################

################################################################################
# BEGIN fix (1 / 44) for 'xccdf_org.ssgproject.content_rule_partition_for_home'
################################################################################
(>&2 echo "Remediating rule 1/44: 'xccdf_org.ssgproject.content_rule_partition_for_home'")
(>&2 echo "FIX FOR THIS RULE 'xccdf_org.ssgproject.content_rule_partition_for_home' IS MISSING!")

# END fix for 'xccdf_org.ssgproject.content_rule_partition_for_home'

################################################################################
# BEGIN fix (2 / 44) for 'xccdf_org.ssgproject.content_rule_partition_for_tmp'
################################################################################
(>&2 echo "Remediating rule 2/44: 'xccdf_org.ssgproject.content_rule_partition_for_tmp'")
(>&2 echo "FIX FOR THIS RULE 'xccdf_org.ssgproject.content_rule_partition_for_tmp' IS MISSING!")

# END fix for 'xccdf_org.ssgproject.content_rule_partition_for_tmp'

################################################################################
```

```
# BEGIN fix (3 / 44) for 'xccdf_org.ssgproject.content_rule_partition_for_var'
#######################################################################
#########
(>&2 echo "Remediating rule 3/44:
'xccdf_org.ssgproject.content_rule_partition_for_var'")
(>&2 echo "FIX FOR THIS RULE 'xccdf_org.ssgproject.content_rule_partition_for_var' IS
MISSING!")

# END fix for 'xccdf_org.ssgproject.content_rule_partition_for_var'

#######################################################################
#########
# BEGIN fix (4 / 44) for 'xccdf_org.ssgproject.content_rule_partition_for_var_log'
#######################################################################
#########
(>&2 echo "Remediating rule 4/44:
'xccdf_org.ssgproject.content_rule_partition_for_var_log'")
(>&2 echo "FIX FOR THIS RULE
'xccdf_org.ssgproject.content_rule_partition_for_var_log' IS MISSING!")

# END fix for 'xccdf_org.ssgproject.content_rule_partition_for_var_log'

#######################################################################
#########
# BEGIN fix (5 / 44) for 'xccdf_org.ssgproject.content_rule_partition_for_var_log_audit'
#######################################################################
#########
(>&2 echo "Remediating rule 5/44:
'xccdf_org.ssgproject.content_rule_partition_for_var_log_audit'")
(>&2 echo "FIX FOR THIS RULE
'xccdf_org.ssgproject.content_rule_partition_for_var_log_audit' IS MISSING!")
```

```
# END fix for 'xccdf_org.ssgproject.content_rule_partition_for_var_log_audit'

################################################################################
# BEGIN fix (6 / 44) for 'xccdf_org.ssgproject.content_rule_package_audit_installed'
################################################################################
(>&2 echo "Remediating rule 6/44: 'xccdf_org.ssgproject.content_rule_package_audit_installed'")
# Remediation is applicable only in certain platforms
if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ]; then

DEBIAN_FRONTEND=noninteractive apt-get install -y "auditd"

else
   >&2 echo 'Remediation is not applicable, nothing was done'
fi

# END fix for 'xccdf_org.ssgproject.content_rule_package_audit_installed'

################################################################################
# BEGIN fix (7 / 44) for 'xccdf_org.ssgproject.content_rule_service_auditd_enabled'
################################################################################
(>&2 echo "Remediating rule 7/44: 'xccdf_org.ssgproject.content_rule_service_auditd_enabled'")
# Remediation is applicable only in certain platforms
```

```
if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ] && { dpkg-query --show --
showformat='${db:Status-Status}\n' 'auditd' 2>/dev/null | grep -q installed; }; then

SYSTEMCTL_EXEC='/usr/bin/systemctl'
"$SYSTEMCTL_EXEC" unmask 'auditd.service'
"$SYSTEMCTL_EXEC" start 'auditd.service'
"$SYSTEMCTL_EXEC" enable 'auditd.service'

else
   >&2 echo 'Remediation is not applicable, nothing was done'
fi

# END fix for 'xccdf_org.ssgproject.content_rule_service_auditd_enabled'

###############################################################################
# BEGIN fix (8 / 44) for 'xccdf_org.ssgproject.content_rule_package_rsyslog_installed'
###############################################################################
(>&2 echo "Remediating rule 8/44:
'xccdf_org.ssgproject.content_rule_package_rsyslog_installed'")
# Remediation is applicable only in certain platforms
if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ]; then

DEBIAN_FRONTEND=noninteractive apt-get install -y "rsyslog"

else
   >&2 echo 'Remediation is not applicable, nothing was done'
fi
```

```
# END fix for 'xccdf_org.ssgproject.content_rule_package_rsyslog_installed'


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

# BEGIN fix (9 / 44) for 'xccdf_org.ssgproject.content_rule_service_rsyslog_enabled'

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

(>&2 echo "Remediating rule 9/44:
'xccdf_org.ssgproject.content_rule_service_rsyslog_enabled'")

# Remediation is applicable only in certain platforms

if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ]; then


SYSTEMCTL_EXEC='/usr/bin/systemctl'

"$SYSTEMCTL_EXEC" unmask 'rsyslog.service'

"$SYSTEMCTL_EXEC" start 'rsyslog.service'

"$SYSTEMCTL_EXEC" enable 'rsyslog.service'


else

   >&2 echo 'Remediation is not applicable, nothing was done'

fi


# END fix for 'xccdf_org.ssgproject.content_rule_service_rsyslog_enabled'


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

# BEGIN fix (10 / 44) for
'xccdf_org.ssgproject.content_rule_rsyslog_files_groupownership'

###############################################################################
```

```
(>&2 echo "Remediating rule 10/44:
'xccdf_org.ssgproject.content_rule_rsyslog_files_groupownership'")

# Remediation is applicable only in certain platforms

if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ]; then


# List of log file paths to be inspected for correct permissions

# * Primarily inspect log file paths listed in /etc/rsyslog.conf

RSYSLOG_ETC_CONFIG="/etc/rsyslog.conf"

# * And also the log file paths listed after rsyslog's $IncludeConfig directive

#   (store the result into array for the case there's shell glob used as value of
IncludeConfig)

readarray -t OLD_INC < <(grep -e "\$IncludeConfig[[:space:]]\+[^[:space:];]\+"
/etc/rsyslog.conf | cut -d ' ' -f 2)

readarray -t RSYSLOG_INCLUDE_CONFIG < <(for INCPATH in "${OLD_INC[@]}"; do eval
printf '%s\\n' "${INCPATH}"; done)

readarray -t NEW_INC < <(awk '/)/{f=0} /include\(/{f=1}
f{nf=gensub("^(include\\(|\\s*)file=\"(\\S+)\".*","\\2",1); if($0!=nf){print nf}}'
/etc/rsyslog.conf)

readarray -t RSYSLOG_INCLUDE < <(for INCPATH in "${NEW_INC[@]}"; do eval printf
'%s\\n' "${INCPATH}"; done)


# Declare an array to hold the final list of different log file paths

declare -a LOG_FILE_PATHS


# Array to hold all rsyslog config entries

RSYSLOG_CONFIGS=()

RSYSLOG_CONFIGS=("${RSYSLOG_ETC_CONFIG}"
"${RSYSLOG_INCLUDE_CONFIG[@]}" "${RSYSLOG_INCLUDE[@]}")


# Get full list of files to be checked
# RSYSLOG_CONFIGS may contain globs such as
# /etc/rsyslog.d/*.conf /etc/rsyslog.d/*.frule
# So, loop over the entries in RSYSLOG_CONFIGS and use find to get the list of included
files.
RSYSLOG_CONFIG_FILES=()
for ENTRY in "${RSYSLOG_CONFIGS[@]}"
do
        # If directory, rsyslog will search for config files in recursively.
        # However, files in hidden sub-directories or hidden files will be ignored.
        if [ -d "${ENTRY}" ]
        then
                readarray -t FINDOUT < <(find "${ENTRY}" -not -path '*/.*' -type f)
                RSYSLOG_CONFIG_FILES+=("${FINDOUT[@]}")
        elif [ -f "${ENTRY}" ]
        then
                RSYSLOG_CONFIG_FILES+=("${ENTRY}")
        else
                echo "Invalid include object: ${ENTRY}"
        fi
done

# Browse each file selected above as containing paths of log files
# ('/etc/rsyslog.conf' and '/etc/rsyslog.d/*.conf' in the default configuration)
for LOG_FILE in "${RSYSLOG_CONFIG_FILES[@]}"
do
        # From each of these files extract just particular log file path(s), thus:
        # * Ignore lines starting with space (' '), comment ('#"), or variable syntax ('$')
characters,
        # * Ignore empty lines,
        # * Strip quotes and closing brackets from paths.
        # * Ignore paths that match /dev|/etc.*\.conf, as those are paths, but likely not log
files
        # * From the remaining valid rows select only fields constituting a log file path
        # Text file column is understood to represent a log file path if and only if all of the
        # following are met:
        # * it contains at least one slash '/' character,
        # * it is preceded by space
        # * it doesn't contain space (' '), colon (':'), and semicolon (';') characters
        # Search log file for path(s) only in case it exists!
        if [[ -f "${LOG_FILE}" ]]
        then
                NORMALIZED_CONFIG_FILE_LINES=$(sed -e "/^[#|$]/d" "${LOG_FILE}")
                LINES_WITH_PATHS=$(grep '[^/]*\s\+\S*/\S\+$' <<<
"${NORMALIZED_CONFIG_FILE_LINES}")
                FILTERED_PATHS=$(awk '{if(NF>=2&&($NF~/^\//||$NF~/^-\//)){sub(/^-
\//,"/",$NF);print $NF}}' <<< "${LINES_WITH_PATHS}")
                CLEANED_PATHS=$(sed -e "s/[\"')]//g; /\\/etc.*\.conf/d; /\\/dev\\//d" <<<
"${FILTERED_PATHS}")
                MATCHED_ITEMS=$(sed -e "/^$/d" <<< "${CLEANED_PATHS}")
                # Since above sed command might return more than one item (delimited
by newline), split
                # the particular matches entries into new array specific for this log file
                readarray -t ARRAY_FOR_LOG_FILE <<< "$MATCHED_ITEMS"
                # Concatenate the two arrays - previous content of $LOG_FILE_PATHS
array with
                # items from newly created array for this log file
                LOG_FILE_PATHS+=("${ARRAY_FOR_LOG_FILE[@]}")
                # Delete the temporary array
                unset ARRAY_FOR_LOG_FILE
        fi
done

# Check for RainerScript action log format which might be also multiline so grep regex is
a bit
# curly:
# extract possibly multiline action omfile expressions
# extract File="logfile" expression
# match only "logfile" expression
for LOG_FILE in "${RSYSLOG_CONFIG_FILES[@]}"
do
        ACTION_OMFILE_LINES=$(grep -ozP "action\s*\(\s*type\s*=\s*\"omfile\"[^\)]*\)"
"${LOG_FILE}")
        OMFILE_LINES=$(echo "${ACTION_OMFILE_LINES}"| grep -aoP
"File\s*=\s*\"([/[:alnum:][:punct:]]*)\"\s*\)")
        LOG_FILE_PATHS+=("$(echo "${OMFILE_LINES}"| grep -oE
"\"([/[:alnum:][:punct:]]*)\""|tr -d "\"")")
done

# Ensure the correct attribute if file exists
FILE_CMD="chgrp"
for LOG_FILE_PATH in "${LOG_FILE_PATHS[@]}"
do
        # Sanity check - if particular $LOG_FILE_PATH is empty string, skip it from further
processing
        if [ -z "$LOG_FILE_PATH" ]
        then
                continue
        fi
        $FILE_CMD "4" "$LOG_FILE_PATH"
done


else
  >&2 echo 'Remediation is not applicable, nothing was done'
fi


# END fix for 'xccdf_org.ssgproject.content_rule_rsyslog_files_groupownership'


###############################################################################
# BEGIN fix (11 / 44) for 'xccdf_org.ssgproject.content_rule_rsyslog_files_ownership'
###############################################################################
(>&2 echo "Remediating rule 11/44: 'xccdf_org.ssgproject.content_rule_rsyslog_files_ownership'")
# Remediation is applicable only in certain platforms
if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ]; then


# List of log file paths to be inspected for correct permissions
# * Primarily inspect log file paths listed in /etc/rsyslog.conf
RSYSLOG_ETC_CONFIG="/etc/rsyslog.conf"
# * And also the log file paths listed after rsyslog's $IncludeConfig directive
#   (store the result into array for the case there's shell glob used as value of IncludeConfig)
readarray -t OLD_INC < <(grep -e "\$IncludeConfig[[:space:]]\+[^[:space:];]\+" /etc/rsyslog.conf | cut -d ' ' -f 2)
readarray -t RSYSLOG_INCLUDE_CONFIG < <(for INCPATH in "${OLD_INC[@]}"; do eval printf '%s\\n' "${INCPATH}"; done)
readarray -t NEW_INC < <(awk '/)/{f=0} /include\(/{f=1}
f{nf=gensub("^(include\\(|\\s*)file=\"(\\S+)\".*","\\2",1); if($0!=nf){print nf}}'
/etc/rsyslog.conf)

readarray -t RSYSLOG_INCLUDE < <(for INCPATH in "${NEW_INC[@]}"; do eval printf
'%s\\n' "${INCPATH}"; done)


# Declare an array to hold the final list of different log file paths

declare -a LOG_FILE_PATHS


# Array to hold all rsyslog config entries

RSYSLOG_CONFIGS=()

RSYSLOG_CONFIGS=("${RSYSLOG_ETC_CONFIG}"
"${RSYSLOG_INCLUDE_CONFIG[@]}" "${RSYSLOG_INCLUDE[@]}")


# Get full list of files to be checked

# RSYSLOG_CONFIGS may contain globs such as

# /etc/rsyslog.d/*.conf /etc/rsyslog.d/*.frule

# So, loop over the entries in RSYSLOG_CONFIGS and use find to get the list of included
files.

RSYSLOG_CONFIG_FILES=()

for ENTRY in "${RSYSLOG_CONFIGS[@]}"

do

        # If directory, rsyslog will search for config files in recursively.

        # However, files in hidden sub-directories or hidden files will be ignored.

        if [ -d "${ENTRY}" ]

        then

                readarray -t FINDOUT < <(find "${ENTRY}" -not -path '*/.*' -type f)

                RSYSLOG_CONFIG_FILES+=("${FINDOUT[@]}")

        elif [ -f "${ENTRY}" ]
```

```
	then
		RSYSLOG_CONFIG_FILES+=("${ENTRY}")
	else
		echo "Invalid include object: ${ENTRY}"
	fi
done

# Browse each file selected above as containing paths of log files
# ('/etc/rsyslog.conf' and '/etc/rsyslog.d/*.conf' in the default configuration)
for LOG_FILE in "${RSYSLOG_CONFIG_FILES[@]}"
do
	# From each of these files extract just particular log file path(s), thus:
	# * Ignore lines starting with space (' '), comment ('#"), or variable syntax ('$')
characters,
	# * Ignore empty lines,
	# * Strip quotes and closing brackets from paths.
	# * Ignore paths that match /dev|/etc.*\.conf, as those are paths, but likely not log
files
	# * From the remaining valid rows select only fields constituting a log file path
	# Text file column is understood to represent a log file path if and only if all of the
	# following are met:
	# * it contains at least one slash '/' character,
	# * it is preceded by space
	# * it doesn't contain space (' '), colon (':'), and semicolon (';') characters
	# Search log file for path(s) only in case it exists!
	if [[ -f "${LOG_FILE}" ]]
	then
		NORMALIZED_CONFIG_FILE_LINES=$(sed -e "/^[#|$]/d" "${LOG_FILE}")
```

```
            LINES_WITH_PATHS=$(grep '[^/]*\s\+\S*/\S\+$' <<<
"${NORMALIZED_CONFIG_FILE_LINES}")

            FILTERED_PATHS=$(awk '{if(NF>=2&&($NF~/^\//||$NF~/^-\//)){sub(/^-
\//,"/",$NF);print $NF}}' <<< "${LINES_WITH_PATHS}")

            CLEANED_PATHS=$(sed -e "s/[\"')]//g; /\\/etc.*\.conf/d; /\\/dev\\//d" <<<
"${FILTERED_PATHS}")

            MATCHED_ITEMS=$(sed -e "/^$/d" <<< "${CLEANED_PATHS}")

            # Since above sed command might return more than one item (delimited
by newline), split

            # the particular matches entries into new array specific for this log file

            readarray -t ARRAY_FOR_LOG_FILE <<< "$MATCHED_ITEMS"

            # Concatenate the two arrays - previous content of $LOG_FILE_PATHS
array with

            # items from newly created array for this log file

            LOG_FILE_PATHS+=("${ARRAY_FOR_LOG_FILE[@]}")

            # Delete the temporary array

            unset ARRAY_FOR_LOG_FILE

      fi

done


# Check for RainerScript action log format which might be also multiline so grep regex is
a bit

# curly:

# extract possibly multiline action omfile expressions

# extract File="logfile" expression

# match only "logfile" expression

for LOG_FILE in "${RSYSLOG_CONFIG_FILES[@]}"

do

      ACTION_OMFILE_LINES=$(grep -ozP "action\s*\(\s*type\s*=\s*\"omfile\"[^\)]*\)"
"${LOG_FILE}")
```

```
	OMFILE_LINES=$(echo "${ACTION_OMFILE_LINES}"| grep -aoP
"File\s*=\s*\"([/[:alnum:][:punct:]]*)\"\s*\)")

	LOG_FILE_PATHS+=("$(echo "${OMFILE_LINES}"| grep -oE
"\"([/[:alnum:][:punct:]]*)\""|tr -d "\"")")

done


# Ensure the correct attribute if file exists

FILE_CMD="chown"

for LOG_FILE_PATH in "${LOG_FILE_PATHS[@]}"

do

	# Sanity check - if particular $LOG_FILE_PATH is empty string, skip it from further
processing

	if [ -z "$LOG_FILE_PATH" ]

	then

		continue

	fi

	$FILE_CMD "104" "$LOG_FILE_PATH"

done


else

  >&2 echo 'Remediation is not applicable, nothing was done'

fi


# END fix for 'xccdf_org.ssgproject.content_rule_rsyslog_files_ownership'


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

# BEGIN fix (12 / 44) for 'xccdf_org.ssgproject.content_rule_rsyslog_files_permissions'
```

```
###############################################################################

(>&2 echo "Remediating rule 12/44: 'xccdf_org.ssgproject.content_rule_rsyslog_files_permissions'")

# Remediation is applicable only in certain platforms

if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ]; then


# List of log file paths to be inspected for correct permissions

# * Primarily inspect log file paths listed in /etc/rsyslog.conf

RSYSLOG_ETC_CONFIG="/etc/rsyslog.conf"

# * And also the log file paths listed after rsyslog's $IncludeConfig directive

#   (store the result into array for the case there's shell glob used as value of IncludeConfig)

readarray -t OLD_INC < <(grep -e "\$IncludeConfig[[:space:]]\+[^[:space:];]\+" /etc/rsyslog.conf | cut -d ' ' -f 2)

readarray -t RSYSLOG_INCLUDE_CONFIG < <(for INCPATH in "${OLD_INC[@]}"; do eval printf '%s\\n' "${INCPATH}"; done)

readarray -t NEW_INC < <(awk '/)/{f=0} /include\(/{f=1} f{nf=gensub("^(include\\(|\\s*)file=\"(\\S+)\".*","\\2",1); if($0!=nf){print nf}}' /etc/rsyslog.conf)

readarray -t RSYSLOG_INCLUDE < <(for INCPATH in "${NEW_INC[@]}"; do eval printf '%s\\n' "${INCPATH}"; done)


# Declare an array to hold the final list of different log file paths

declare -a LOG_FILE_PATHS


# Array to hold all rsyslog config entries

RSYSLOG_CONFIGS=()

RSYSLOG_CONFIGS=("${RSYSLOG_ETC_CONFIG}" "${RSYSLOG_INCLUDE_CONFIG[@]}" "${RSYSLOG_INCLUDE[@]}")
```

```
# Get full list of files to be checked

# RSYSLOG_CONFIGS may contain globs such as

# /etc/rsyslog.d/*.conf /etc/rsyslog.d/*.frule

# So, loop over the entries in RSYSLOG_CONFIGS and use find to get the list of included
files.

RSYSLOG_CONFIG_FILES=()

for ENTRY in "${RSYSLOG_CONFIGS[@]}"

do

	# If directory, rsyslog will search for config files in recursively.

	# However, files in hidden sub-directories or hidden files will be ignored.

	if [ -d "${ENTRY}" ]

	then

		readarray -t FINDOUT < <(find "${ENTRY}" -not -path '*/.*' -type f)

		RSYSLOG_CONFIG_FILES+=("${FINDOUT[@]}")

	elif [ -f "${ENTRY}" ]

	then

		RSYSLOG_CONFIG_FILES+=("${ENTRY}")

	else

		echo "Invalid include object: ${ENTRY}"

	fi

done


# Browse each file selected above as containing paths of log files

# ('/etc/rsyslog.conf' and '/etc/rsyslog.d/*.conf' in the default configuration)

for LOG_FILE in "${RSYSLOG_CONFIG_FILES[@]}"

do

	# From each of these files extract just particular log file path(s), thus:

	# * Ignore lines starting with space (' '), comment ('#"), or variable syntax ('$')
characters,
    # * Ignore empty lines,
    # * Strip quotes and closing brackets from paths.
    # * Ignore paths that match /dev|/etc.*\.conf, as those are paths, but likely not log
files
    # * From the remaining valid rows select only fields constituting a log file path
    # Text file column is understood to represent a log file path if and only if all of the
    # following are met:
    # * it contains at least one slash '/' character,
    # * it is preceded by space
    # * it doesn't contain space (' '), colon (':'), and semicolon (';') characters
    # Search log file for path(s) only in case it exists!
    if [[ -f "${LOG_FILE}" ]]
    then
        NORMALIZED_CONFIG_FILE_LINES=$(sed -e "/^[#|$]/d" "${LOG_FILE}")
        LINES_WITH_PATHS=$(grep '[^/]*\s\+\S*/\S\+$' <<<
"${NORMALIZED_CONFIG_FILE_LINES}")
        FILTERED_PATHS=$(awk '{if(NF>=2&&($NF~/^\//||$NF~/^-\//)){sub(/^-
\//,"/",$NF);print $NF}}' <<< "${LINES_WITH_PATHS}")
        CLEANED_PATHS=$(sed -e "s/[\"')]//g; /\\/etc.*\.conf/d; /\\/dev\\//d" <<<
"${FILTERED_PATHS}")
        MATCHED_ITEMS=$(sed -e "/^$/d" <<< "${CLEANED_PATHS}")
        # Since above sed command might return more than one item (delimited
by newline), split
        # the particular matches entries into new array specific for this log file
        readarray -t ARRAY_FOR_LOG_FILE <<< "$MATCHED_ITEMS"
        # Concatenate the two arrays - previous content of $LOG_FILE_PATHS
array with
        # items from newly created array for this log file
        LOG_FILE_PATHS+=("${ARRAY_FOR_LOG_FILE[@]}")
        # Delete the temporary array
```

```
			unset ARRAY_FOR_LOG_FILE
		fi
done

# Check for RainerScript action log format which might be also multiline so grep regex is a bit
# curly:
# extract possibly multiline action omfile expressions
# extract File="logfile" expression
# match only "logfile" expression
for LOG_FILE in "${RSYSLOG_CONFIG_FILES[@]}"
do
		ACTION_OMFILE_LINES=$(grep -ozP "action\s*\(\s*type\s*=\s*\"omfile\"[^\)]*\)" "${LOG_FILE}")
		OMFILE_LINES=$(echo "${ACTION_OMFILE_LINES}"| grep -aoP "File\s*=\s*\"([/[:alnum:][:punct:]]*)\"\s*\)")
		LOG_FILE_PATHS+=("$(echo "${OMFILE_LINES}"| grep -oE "\"([/[:alnum:][:punct:]]*)\""|tr -d "\"")")
done

# Ensure the correct attribute if file exists
FILE_CMD="chmod"
for LOG_FILE_PATH in "${LOG_FILE_PATHS[@]}"
do
		# Sanity check - if particular $LOG_FILE_PATH is empty string, skip it from further processing
		if [ -z "$LOG_FILE_PATH" ]
		then
			continue
```

```
        fi
        $FILE_CMD "0640" "$LOG_FILE_PATH"
done

else
  >&2 echo 'Remediation is not applicable, nothing was done'
fi

# END fix for 'xccdf_org.ssgproject.content_rule_rsyslog_files_permissions'

###############################################################################
# BEGIN fix (13 / 44) for 'xccdf_org.ssgproject.content_rule_ensure_logrotate_activated'
###############################################################################
(>&2 echo "Remediating rule 13/44:
'xccdf_org.ssgproject.content_rule_ensure_logrotate_activated'")
# Remediation is applicable only in certain platforms
if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ]; then

LOGROTATE_CONF_FILE="/etc/logrotate.conf"
CRON_DAILY_LOGROTATE_FILE="/etc/cron.daily/logrotate"

# daily rotation is configured
grep -q "^daily$" $LOGROTATE_CONF_FILE|| echo "daily" >> $LOGROTATE_CONF_FILE

# remove any line configuring weekly, monthly or yearly rotation
sed -i '/^\s*\(weekly\|monthly\|yearly\).*$/d' $LOGROTATE_CONF_FILE
```

```
# configure cron.daily if not already

if ! grep -q
"^[[:space:]]*/usr/sbin/logrotate[[:alnum:][:blank:][:punct:]]*$LOGROTATE_CONF_FILE
$" $CRON_DAILY_LOGROTATE_FILE; then

        echo "#!/bin/sh" > $CRON_DAILY_LOGROTATE_FILE

        echo "/usr/sbin/logrotate $LOGROTATE_CONF_FILE" >>
$CRON_DAILY_LOGROTATE_FILE

fi


else

   >&2 echo 'Remediation is not applicable, nothing was done'

fi


# END fix for 'xccdf_org.ssgproject.content_rule_ensure_logrotate_activated'


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

# BEGIN fix (14 / 44) for
'xccdf_org.ssgproject.content_rule_file_permissions_systemmap'

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

(>&2 echo "Remediating rule 14/44:
'xccdf_org.ssgproject.content_rule_file_permissions_systemmap'")

(>&2 echo "FIX FOR THIS RULE
'xccdf_org.ssgproject.content_rule_file_permissions_systemmap' IS MISSING!")


# END fix for 'xccdf_org.ssgproject.content_rule_file_permissions_systemmap'


###############################################################################
#########
```

```
# BEGIN fix (15 / 44) for
'xccdf_org.ssgproject.content_rule_sysctl_fs_protected_hardlinks'

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

(>&2 echo "Remediating rule 15/44:
'xccdf_org.ssgproject.content_rule_sysctl_fs_protected_hardlinks'")

# Remediation is applicable only in certain platforms

if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ]; then


# Comment out any occurrences of fs.protected_hardlinks from /etc/sysctl.d/*.conf files


for f in /etc/sysctl.d/*.conf /run/sysctl.d/*.conf /usr/local/lib/sysctl.d/*.conf
/usr/lib/sysctl.d/*.conf; do


  matching_list=$(grep -P '^(?!#).*[\s]*fs.protected_hardlinks.*$' $f | uniq )

  if ! test -z "$matching_list"; then

    while IFS= read -r entry; do

      escaped_entry=$(sed -e 's|/|\\/|g' <<< "$entry")

      # comment out "fs.protected_hardlinks" matches to preserve user data

      sed -i "s/^${escaped_entry}$/# &/g" $f

    done <<< "$matching_list"

  fi

done


#

# Set runtime for fs.protected_hardlinks

#

/sbin/sysctl -q -n -w fs.protected_hardlinks="1"
```

```
#
# If fs.protected_hardlinks present in /etc/sysctl.conf, change value to "1"
#         else, add "fs.protected_hardlinks = 1" to /etc/sysctl.conf
#
# Test if the config_file is a symbolic link. If so, use --follow-symlinks with sed.
# Otherwise, regular sed command will do.
sed_command=('sed' '-i')
if test -L "/etc/sysctl.conf"; then
  sed_command+=('--follow-symlinks')
fi

# Strip any search characters in the key arg so that the key can be replaced without
# adding any search characters to the config file.
stripped_key=$(sed 's/[\^=\$,;+]*//g' <<< "^fs.protected_hardlinks")

# shellcheck disable=SC2059
printf -v formatted_output "%s = %s" "$stripped_key" "1"

# If the key exists, change it. Otherwise, add it to the config_file.
# We search for the key string followed by a word boundary (matched by \>),
# so if we search for 'setting', 'setting2' won't match.
if LC_ALL=C grep -q -m 1 -i -e "^fs.protected_hardlinks\\>" "/etc/sysctl.conf"; then
  escaped_formatted_output=$(sed -e 's|/|\\/|g' <<< "$formatted_output")
  "${sed_command[@]}"
"s/^fs.protected_hardlinks\\>.*/$escaped_formatted_output/gi" "/etc/sysctl.conf"
else
  # \n is precaution for case where file ends without trailing newline
  printf '%s\n' "$formatted_output" >> "/etc/sysctl.conf"
fi


else
  >&2 echo 'Remediation is not applicable, nothing was done'
fi


# END fix for 'xccdf_org.ssgproject.content_rule_sysctl_fs_protected_hardlinks'


#########################################################################
#########
# BEGIN fix (16 / 44) for
'xccdf_org.ssgproject.content_rule_sysctl_fs_protected_symlinks'
#########################################################################
#########
(>&2 echo "Remediating rule 16/44:
'xccdf_org.ssgproject.content_rule_sysctl_fs_protected_symlinks'")
# Remediation is applicable only in certain platforms
if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ]; then


# Comment out any occurrences of fs.protected_symlinks from /etc/sysctl.d/*.conf files


for f in /etc/sysctl.d/*.conf /run/sysctl.d/*.conf /usr/local/lib/sysctl.d/*.conf
/usr/lib/sysctl.d/*.conf; do


 matching_list=$(grep -P '^(?!#).*[\s]*fs.protected_symlinks.*$' $f | uniq )
 if ! test -z "$matching_list"; then
  while IFS= read -r entry; do
   escaped_entry=$(sed -e 's|/|\\/|g' <<< "$entry")
```

```
      # comment out "fs.protected_symlinks" matches to preserve user data
      sed -i "s/^${escaped_entry}$/# &/g" $f
    done <<< "$matching_list"
  fi
done

#
# Set runtime for fs.protected_symlinks
#
/sbin/sysctl -q -n -w fs.protected_symlinks="1"

#
# If fs.protected_symlinks present in /etc/sysctl.conf, change value to "1"
#         else, add "fs.protected_symlinks = 1" to /etc/sysctl.conf
#
# Test if the config_file is a symbolic link. If so, use --follow-symlinks with sed.
# Otherwise, regular sed command will do.
sed_command=('sed' '-i')
if test -L "/etc/sysctl.conf"; then
    sed_command+=('--follow-symlinks')
fi

# Strip any search characters in the key arg so that the key can be replaced without
# adding any search characters to the config file.
stripped_key=$(sed 's/[\^=\$,;+]*//g' <<< "^fs.protected_symlinks")

# shellcheck disable=SC2059
printf -v formatted_output "%s = %s" "$stripped_key" "1"
```

```
# If the key exists, change it. Otherwise, add it to the config_file.

# We search for the key string followed by a word boundary (matched by \>),

# so if we search for 'setting', 'setting2' won't match.

if LC_ALL=C grep -q -m 1 -i -e "^fs.protected_symlinks\\>" "/etc/sysctl.conf"; then

   escaped_formatted_output=$(sed -e 's|/|\\/|g' <<< "$formatted_output")

   "${sed_command[@]}"
"s/^fs.protected_symlinks\\>.*/$escaped_formatted_output/gi" "/etc/sysctl.conf"

else

   # \n is precaution for case where file ends without trailing newline


   printf '%s\n' "$formatted_output" >> "/etc/sysctl.conf"

fi


else

   >&2 echo 'Remediation is not applicable, nothing was done'

fi


# END fix for 'xccdf_org.ssgproject.content_rule_sysctl_fs_protected_symlinks'


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

# BEGIN fix (17 / 44) for 'xccdf_org.ssgproject.content_rule_file_groupowner_etc_group'

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

(>&2 echo "Remediating rule 17/44:
'xccdf_org.ssgproject.content_rule_file_groupowner_etc_group'")
```

```
chgrp 0 /etc/group

# END fix for 'xccdf_org.ssgproject.content_rule_file_groupowner_etc_group'

#######################################################################
#########
# BEGIN fix (18 / 44) for
'xccdf_org.ssgproject.content_rule_file_groupowner_etc_gshadow'
#######################################################################
#########
(>&2 echo "Remediating rule 18/44:
'xccdf_org.ssgproject.content_rule_file_groupowner_etc_gshadow'")



chgrp 42 /etc/gshadow

# END fix for 'xccdf_org.ssgproject.content_rule_file_groupowner_etc_gshadow'

#######################################################################
#########
# BEGIN fix (19 / 44) for
'xccdf_org.ssgproject.content_rule_file_groupowner_etc_passwd'
#######################################################################
#########
(>&2 echo "Remediating rule 19/44:
'xccdf_org.ssgproject.content_rule_file_groupowner_etc_passwd'")
```

```
chgrp 0 /etc/passwd

# END fix for 'xccdf_org.ssgproject.content_rule_file_groupowner_etc_passwd'

#######################################################################
#########
# BEGIN fix (20 / 44) for
'xccdf_org.ssgproject.content_rule_file_groupowner_etc_shadow'
#######################################################################
#########
(>&2 echo "Remediating rule 20/44:
'xccdf_org.ssgproject.content_rule_file_groupowner_etc_shadow'")



chgrp 42 /etc/shadow

# END fix for 'xccdf_org.ssgproject.content_rule_file_groupowner_etc_shadow'

#######################################################################
#########
# BEGIN fix (21 / 44) for 'xccdf_org.ssgproject.content_rule_file_owner_etc_group'
#######################################################################
#########
(>&2 echo "Remediating rule 21/44:
'xccdf_org.ssgproject.content_rule_file_owner_etc_group'")
```

```
chown 0 /etc/group

# END fix for 'xccdf_org.ssgproject.content_rule_file_owner_etc_group'

#######################################################################
#########
# BEGIN fix (22 / 44) for 'xccdf_org.ssgproject.content_rule_file_owner_etc_gshadow'
#######################################################################
#########
(>&2 echo "Remediating rule 22/44:
'xccdf_org.ssgproject.content_rule_file_owner_etc_gshadow'")



chown 0 /etc/gshadow

# END fix for 'xccdf_org.ssgproject.content_rule_file_owner_etc_gshadow'

#######################################################################
#########
# BEGIN fix (23 / 44) for 'xccdf_org.ssgproject.content_rule_file_owner_etc_passwd'
#######################################################################
#########
(>&2 echo "Remediating rule 23/44:
'xccdf_org.ssgproject.content_rule_file_owner_etc_passwd'")



chown 0 /etc/passwd
```

```
# END fix for 'xccdf_org.ssgproject.content_rule_file_owner_etc_passwd'

###############################################################################
# BEGIN fix (24 / 44) for 'xccdf_org.ssgproject.content_rule_file_owner_etc_shadow'
###############################################################################
(>&2 echo "Remediating rule 24/44:
'xccdf_org.ssgproject.content_rule_file_owner_etc_shadow'")



chown 0 /etc/shadow

# END fix for 'xccdf_org.ssgproject.content_rule_file_owner_etc_shadow'

###############################################################################
# BEGIN fix (25 / 44) for 'xccdf_org.ssgproject.content_rule_file_permissions_etc_group'
###############################################################################
(>&2 echo "Remediating rule 25/44:
'xccdf_org.ssgproject.content_rule_file_permissions_etc_group'")




chmod u-xs,g-xws,o-xwt /etc/group

# END fix for 'xccdf_org.ssgproject.content_rule_file_permissions_etc_group'
```

```
###############################################################################
# BEGIN fix (26 / 44) for 'xccdf_org.ssgproject.content_rule_file_permissions_etc_gshadow'
###############################################################################
(>&2 echo "Remediating rule 26/44: 'xccdf_org.ssgproject.content_rule_file_permissions_etc_gshadow'")




chmod u-xs,g-xws,o-xwrt /etc/gshadow


# END fix for 'xccdf_org.ssgproject.content_rule_file_permissions_etc_gshadow'


###############################################################################
# BEGIN fix (27 / 44) for 'xccdf_org.ssgproject.content_rule_file_permissions_etc_passwd'
###############################################################################
(>&2 echo "Remediating rule 27/44: 'xccdf_org.ssgproject.content_rule_file_permissions_etc_passwd'")




chmod u-xs,g-xws,o-xwt /etc/passwd
```

```
# END fix for 'xccdf_org.ssgproject.content_rule_file_permissions_etc_passwd'

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

# BEGIN fix (28 / 44) for 'xccdf_org.ssgproject.content_rule_file_permissions_etc_shadow'

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

(>&2 echo "Remediating rule 28/44: 'xccdf_org.ssgproject.content_rule_file_permissions_etc_shadow'")




chmod u-xs,g-xws,o-xwrt /etc/shadow


# END fix for 'xccdf_org.ssgproject.content_rule_file_permissions_etc_shadow'


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

# BEGIN fix (29 / 44) for 'xccdf_org.ssgproject.content_rule_sysctl_fs_suid_dumpable'

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

(>&2 echo "Remediating rule 29/44: 'xccdf_org.ssgproject.content_rule_sysctl_fs_suid_dumpable'")

# Remediation is applicable only in certain platforms

if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ]; then


# Comment out any occurrences of fs.suid_dumpable from /etc/sysctl.d/*.conf files
```

```
for f in /etc/sysctl.d/*.conf /run/sysctl.d/*.conf /usr/local/lib/sysctl.d/*.conf
/usr/lib/sysctl.d/*.conf; do

  matching_list=$(grep -P '^(?!#).*[\s]*fs.suid_dumpable.*$' $f | uniq )
  if ! test -z "$matching_list"; then
    while IFS= read -r entry; do
      escaped_entry=$(sed -e 's|/|\\/|g' <<< "$entry")
      # comment out "fs.suid_dumpable" matches to preserve user data
      sed -i "s/^${escaped_entry}$/# &/g" $f
    done <<< "$matching_list"
  fi
done

#
# Set runtime for fs.suid_dumpable
#
/sbin/sysctl -q -n -w fs.suid_dumpable="0"

#
# If fs.suid_dumpable present in /etc/sysctl.conf, change value to "0"
#         else, add "fs.suid_dumpable = 0" to /etc/sysctl.conf
#
# Test if the config_file is a symbolic link. If so, use --follow-symlinks with sed.
# Otherwise, regular sed command will do.
sed_command=('sed' '-i')
if test -L "/etc/sysctl.conf"; then
    sed_command+=('--follow-symlinks')
```

```
fi


# Strip any search characters in the key arg so that the key can be replaced without
# adding any search characters to the config file.
stripped_key=$(sed 's/[\^=\$,;+]*//g' <<< "^fs.suid_dumpable")


# shellcheck disable=SC2059
printf -v formatted_output "%s = %s" "$stripped_key" "0"


# If the key exists, change it. Otherwise, add it to the config_file.
# We search for the key string followed by a word boundary (matched by \>),
# so if we search for 'setting', 'setting2' won't match.
if LC_ALL=C grep -q -m 1 -i -e "^fs.suid_dumpable\\>" "/etc/sysctl.conf"; then
  escaped_formatted_output=$(sed -e 's|/|\\/|g' <<< "$formatted_output")
  "${sed_command[@]}" "s/^fs.suid_dumpable\\>.*/$escaped_formatted_output/gi"
"/etc/sysctl.conf"
else
  # \n is precaution for case where file ends without trailing newline


  printf '%s\n' "$formatted_output" >> "/etc/sysctl.conf"
fi


else
  >&2 echo 'Remediation is not applicable, nothing was done'
fi


# END fix for 'xccdf_org.ssgproject.content_rule_sysctl_fs_suid_dumpable'
```

```
###############################################################################
# BEGIN fix (30 / 44) for 'xccdf_org.ssgproject.content_rule_sysctl_kernel_randomize_va_space'
###############################################################################
(>&2 echo "Remediating rule 30/44: 'xccdf_org.ssgproject.content_rule_sysctl_kernel_randomize_va_space'")
# Remediation is applicable only in certain platforms
if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ]; then

# Comment out any occurrences of kernel.randomize_va_space from /etc/sysctl.d/*.conf files

for f in /etc/sysctl.d/*.conf /run/sysctl.d/*.conf /usr/local/lib/sysctl.d/*.conf /usr/lib/sysctl.d/*.conf; do

  matching_list=$(grep -P '^(?!#).*[\s]*kernel.randomize_va_space.*$' $f | uniq )
  if ! test -z "$matching_list"; then
    while IFS= read -r entry; do
      escaped_entry=$(sed -e 's|/|\\/|g' <<< "$entry")
      # comment out "kernel.randomize_va_space" matches to preserve user data
      sed -i "s/^${escaped_entry}$/# &/g" $f
    done <<< "$matching_list"
  fi
done

#
# Set runtime for kernel.randomize_va_space
#
```

```
/sbin/sysctl -q -n -w kernel.randomize_va_space="2"

#
# If kernel.randomize_va_space present in /etc/sysctl.conf, change value to "2"
#         else, add "kernel.randomize_va_space = 2" to /etc/sysctl.conf
#
# Test if the config_file is a symbolic link. If so, use --follow-symlinks with sed.
# Otherwise, regular sed command will do.
sed_command=('sed' '-i')
if test -L "/etc/sysctl.conf"; then
  sed_command+=('--follow-symlinks')
fi

# Strip any search characters in the key arg so that the key can be replaced without
# adding any search characters to the config file.
stripped_key=$(sed 's/[\^=\$,;+]*//g' <<< "^kernel.randomize_va_space")

# shellcheck disable=SC2059
printf -v formatted_output "%s = %s" "$stripped_key" "2"

# If the key exists, change it. Otherwise, add it to the config_file.
# We search for the key string followed by a word boundary (matched by \>),
# so if we search for 'setting', 'setting2' won't match.
if LC_ALL=C grep -q -m 1 -i -e "^kernel.randomize_va_space\\>" "/etc/sysctl.conf"; then
  escaped_formatted_output=$(sed -e 's|/|\\/|g' <<< "$formatted_output")
  "${sed_command[@]}"
"s/^kernel.randomize_va_space\\>.*/$escaped_formatted_output/gi" "/etc/sysctl.conf"
else
```

```
    # \n is precaution for case where file ends without trailing newline

    printf '%s\n' "$formatted_output" >> "/etc/sysctl.conf"
fi

else
    >&2 echo 'Remediation is not applicable, nothing was done'
fi

# END fix for 'xccdf_org.ssgproject.content_rule_sysctl_kernel_randomize_va_space'

###############################################################################
# BEGIN fix (31 / 44) for 'xccdf_org.ssgproject.content_rule_service_apport_disabled'
###############################################################################
(>&2 echo "Remediating rule 31/44: 'xccdf_org.ssgproject.content_rule_service_apport_disabled'")



SYSTEMCTL_EXEC='/usr/bin/systemctl'
"$SYSTEMCTL_EXEC" stop 'apport.service'
"$SYSTEMCTL_EXEC" disable 'apport.service'
"$SYSTEMCTL_EXEC" mask 'apport.service'
# Disable socket activation if we have a unit file for it
if "$SYSTEMCTL_EXEC" -q list-unit-files apport.socket; then
    "$SYSTEMCTL_EXEC" stop 'apport.socket'
    "$SYSTEMCTL_EXEC" mask 'apport.socket'
fi
```

```
# The service may not be running because it has been started and failed,
# so let's reset the state so OVAL checks pass.
# Service should be 'inactive', not 'failed' after reboot though.
"$SYSTEMCTL_EXEC" reset-failed 'apport.service' || true

# END fix for 'xccdf_org.ssgproject.content_rule_service_apport_disabled'

###############################################################################
# BEGIN fix (32 / 44) for 'xccdf_org.ssgproject.content_rule_package_cron_installed'
###############################################################################
(>&2 echo "Remediating rule 32/44:
'xccdf_org.ssgproject.content_rule_package_cron_installed'")
# Remediation is applicable only in certain platforms
if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ]; then

DEBIAN_FRONTEND=noninteractive apt-get install -y "cron"

else
   >&2 echo 'Remediation is not applicable, nothing was done'
fi

# END fix for 'xccdf_org.ssgproject.content_rule_package_cron_installed'

###############################################################################
# BEGIN fix (33 / 44) for 'xccdf_org.ssgproject.content_rule_service_cron_enabled'
```

```
###############################################################################
(>&2 echo "Remediating rule 33/44: 'xccdf_org.ssgproject.content_rule_service_cron_enabled'")
# Remediation is applicable only in certain platforms
if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ]; then

SYSTEMCTL_EXEC='/usr/bin/systemctl'
"$SYSTEMCTL_EXEC" unmask 'cron.service'
"$SYSTEMCTL_EXEC" start 'cron.service'
"$SYSTEMCTL_EXEC" enable 'cron.service'

else
    >&2 echo 'Remediation is not applicable, nothing was done'
fi

# END fix for 'xccdf_org.ssgproject.content_rule_service_cron_enabled'

###############################################################################
# BEGIN fix (34 / 44) for 'xccdf_org.ssgproject.content_rule_package_inetutils-telnetd_removed'
###############################################################################
(>&2 echo "Remediating rule 34/44: 'xccdf_org.ssgproject.content_rule_package_inetutils-telnetd_removed'")

# CAUTION: This remediation script will remove inetutils-telnetd
#	   from the system, and may remove any packages
#	   that depend on inetutils-telnetd. Execute this
```

```
#	remediation AFTER testing on a non-production
#	system!

DEBIAN_FRONTEND=noninteractive apt-get remove -y "inetutils-telnetd"

# END fix for 'xccdf_org.ssgproject.content_rule_package_inetutils-telnetd_removed'

##########################################################################
#########
# BEGIN fix (35 / 44) for 'xccdf_org.ssgproject.content_rule_package_nis_removed'
##########################################################################
#########
(>&2 echo "Remediating rule 35/44:
'xccdf_org.ssgproject.content_rule_package_nis_removed'")

# CAUTION: This remediation script will remove nis
#	from the system, and may remove any packages
#	that depend on nis. Execute this
#	remediation AFTER testing on a non-production
#	system!

DEBIAN_FRONTEND=noninteractive apt-get remove -y "nis"

# END fix for 'xccdf_org.ssgproject.content_rule_package_nis_removed'

##########################################################################
#########
# BEGIN fix (36 / 44) for 'xccdf_org.ssgproject.content_rule_package_ntpdate_removed'
```

```
###############################################################################
(>&2 echo "Remediating rule 36/44:
'xccdf_org.ssgproject.content_rule_package_ntpdate_removed'")


# CAUTION: This remediation script will remove ntpdate
#	from the system, and may remove any packages
#	that depend on ntpdate. Execute this
#	remediation AFTER testing on a non-production
#	system!


DEBIAN_FRONTEND=noninteractive apt-get remove -y "ntpdate"


# END fix for 'xccdf_org.ssgproject.content_rule_package_ntpdate_removed'


###############################################################################
# BEGIN fix (37 / 44) for 'xccdf_org.ssgproject.content_rule_package_telnetd-
ssl_removed'
###############################################################################
(>&2 echo "Remediating rule 37/44:
'xccdf_org.ssgproject.content_rule_package_telnetd-ssl_removed'")


# CAUTION: This remediation script will remove telnetd-ssl
#	from the system, and may remove any packages
#	that depend on telnetd-ssl. Execute this
#	remediation AFTER testing on a non-production
#	system!
```

```
DEBIAN_FRONTEND=noninteractive apt-get remove -y "telnetd-ssl"


# END fix for 'xccdf_org.ssgproject.content_rule_package_telnetd-ssl_removed'


##############################################################################
##########
# BEGIN fix (38 / 44) for 'xccdf_org.ssgproject.content_rule_package_telnetd_removed'
##############################################################################
##########
(>&2 echo "Remediating rule 38/44:
'xccdf_org.ssgproject.content_rule_package_telnetd_removed'")


# CAUTION: This remediation script will remove telnetd
#	   from the system, and may remove any packages
#	   that depend on telnetd. Execute this
#	   remediation AFTER testing on a non-production
#	   system!


DEBIAN_FRONTEND=noninteractive apt-get remove -y "telnetd"


# END fix for 'xccdf_org.ssgproject.content_rule_package_telnetd_removed'


##############################################################################
##########
# BEGIN fix (39 / 44) for
'xccdf_org.ssgproject.content_rule_package_timesyncd_installed'
##############################################################################
##########
(>&2 echo "Remediating rule 39/44:
'xccdf_org.ssgproject.content_rule_package_timesyncd_installed'")
```

```
# Remediation is applicable only in certain platforms
if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ]; then

DEBIAN_FRONTEND=noninteractive apt-get install -y "systemd-timesyncd"

else
   >&2 echo 'Remediation is not applicable, nothing was done'
fi

# END fix for 'xccdf_org.ssgproject.content_rule_package_timesyncd_installed'

###############################################################################
# BEGIN fix (40 / 44) for 'xccdf_org.ssgproject.content_rule_service_timesyncd_enabled'
###############################################################################
(>&2 echo "Remediating rule 40/44: 'xccdf_org.ssgproject.content_rule_service_timesyncd_enabled'")
# Remediation is applicable only in certain platforms
if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ] && { ( ! ( dpkg-query --show --showformat='${db:Status-Status}\n' 'chrony' 2>/dev/null | grep -q installed ) && ! ( dpkg-query --show --showformat='${db:Status-Status}\n' 'ntp' 2>/dev/null | grep -q installed ) ); }; then

SYSTEMCTL_EXEC='/usr/bin/systemctl'
"$SYSTEMCTL_EXEC" unmask 'systemd-timesyncd.service'
"$SYSTEMCTL_EXEC" start 'systemd-timesyncd.service'
"$SYSTEMCTL_EXEC" enable 'systemd-timesyncd.service'

else
```

```
    >&2 echo 'Remediation is not applicable, nothing was done'
fi


# END fix for 'xccdf_org.ssgproject.content_rule_service_timesyncd_enabled'

###############################################################################
# BEGIN fix (41 / 44) for 'xccdf_org.ssgproject.content_rule_sshd_set_keepalive'
###############################################################################
(>&2 echo "Remediating rule 41/44: 'xccdf_org.ssgproject.content_rule_sshd_set_keepalive'")
# Remediation is applicable only in certain platforms
if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ]; then

var_sshd_set_keepalive='0'



if [ -e "/etc/ssh/sshd_config" ] ; then

    LC_ALL=C sed -i "/^\s*ClientAliveCountMax\s\+/Id" "/etc/ssh/sshd_config"
else
    touch "/etc/ssh/sshd_config"
fi
# make sure file has newline at the end
sed -i -e '$a\' "/etc/ssh/sshd_config"

cp "/etc/ssh/sshd_config" "/etc/ssh/sshd_config.bak"
# Insert before the line matching the regex '^Match'.
```

```
line_number="$(LC_ALL=C grep -n "^Match" "/etc/ssh/sshd_config.bak" | LC_ALL=C sed
's/:.*//g')"

if [ -z "$line_number" ]; then

   # There was no match of '^Match', insert at

   # the end of the file.

   printf '%s\n' "ClientAliveCountMax $var_sshd_set_keepalive" >>
"/etc/ssh/sshd_config"

else

   head -n "$(( line_number - 1 ))" "/etc/ssh/sshd_config.bak" > "/etc/ssh/sshd_config"

   printf '%s\n' "ClientAliveCountMax $var_sshd_set_keepalive" >>
"/etc/ssh/sshd_config"

   tail -n "+$(( line_number ))" "/etc/ssh/sshd_config.bak" >> "/etc/ssh/sshd_config"

fi

# Clean up after ourselves.

rm "/etc/ssh/sshd_config.bak"


else

   >&2 echo 'Remediation is not applicable, nothing was done'

fi


# END fix for 'xccdf_org.ssgproject.content_rule_sshd_set_keepalive'


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

# BEGIN fix (42 / 44) for 'xccdf_org.ssgproject.content_rule_sshd_set_idle_timeout'

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

(>&2 echo "Remediating rule 42/44:
'xccdf_org.ssgproject.content_rule_sshd_set_idle_timeout'")

# Remediation is applicable only in certain platforms
```

```
if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ]; then

sshd_idle_timeout_value='300'



if [ -e "/etc/ssh/sshd_config" ] ; then

   LC_ALL=C sed -i "/^\s*ClientAliveInterval\s\+/Id" "/etc/ssh/sshd_config"
else
   touch "/etc/ssh/sshd_config"
fi
# make sure file has newline at the end
sed -i -e '$a\' "/etc/ssh/sshd_config"

cp "/etc/ssh/sshd_config" "/etc/ssh/sshd_config.bak"
# Insert before the line matching the regex '^Match'.
line_number="$(LC_ALL=C grep -n "^Match" "/etc/ssh/sshd_config.bak" | LC_ALL=C sed
's/:.*//g')"
if [ -z "$line_number" ]; then
   # There was no match of '^Match', insert at
   # the end of the file.
   printf '%s\n' "ClientAliveInterval $sshd_idle_timeout_value" >> "/etc/ssh/sshd_config"
else
   head -n "$(( line_number - 1 ))" "/etc/ssh/sshd_config.bak" > "/etc/ssh/sshd_config"
   printf '%s\n' "ClientAliveInterval $sshd_idle_timeout_value" >> "/etc/ssh/sshd_config"
   tail -n "+$(( line_number ))" "/etc/ssh/sshd_config.bak" >> "/etc/ssh/sshd_config"
fi
# Clean up after ourselves.
```

```
rm "/etc/ssh/sshd_config.bak"


else
   >&2 echo 'Remediation is not applicable, nothing was done'
fi


# END fix for 'xccdf_org.ssgproject.content_rule_sshd_set_idle_timeout'


###############################################################################
# BEGIN fix (43 / 44) for 'xccdf_org.ssgproject.content_rule_sshd_disable_empty_passwords'
###############################################################################
(>&2 echo "Remediating rule 43/44: 'xccdf_org.ssgproject.content_rule_sshd_disable_empty_passwords'")
# Remediation is applicable only in certain platforms
if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ]; then


if [ -e "/etc/ssh/sshd_config" ] ; then


   LC_ALL=C sed -i "/^\s*PermitEmptyPasswords\s\+/Id" "/etc/ssh/sshd_config"
else
   touch "/etc/ssh/sshd_config"
fi
# make sure file has newline at the end
sed -i -e '$a\' "/etc/ssh/sshd_config"


cp "/etc/ssh/sshd_config" "/etc/ssh/sshd_config.bak"
```

```
# Insert before the line matching the regex '^Match'.

line_number="$(LC_ALL=C grep -n "^Match" "/etc/ssh/sshd_config.bak" | LC_ALL=C sed 's/:.*//g')"

if [ -z "$line_number" ]; then

    # There was no match of '^Match', insert at

    # the end of the file.

    printf '%s\n' "PermitEmptyPasswords no" >> "/etc/ssh/sshd_config"

else

    head -n "$(( line_number - 1 ))" "/etc/ssh/sshd_config.bak" > "/etc/ssh/sshd_config"

    printf '%s\n' "PermitEmptyPasswords no" >> "/etc/ssh/sshd_config"

    tail -n "+$(( line_number ))" "/etc/ssh/sshd_config.bak" >> "/etc/ssh/sshd_config"

fi

# Clean up after ourselves.

rm "/etc/ssh/sshd_config.bak"


else

    >&2 echo 'Remediation is not applicable, nothing was done'

fi


# END fix for 'xccdf_org.ssgproject.content_rule_sshd_disable_empty_passwords'


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

# BEGIN fix (44 / 44) for 'xccdf_org.ssgproject.content_rule_sshd_disable_root_login'

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

(>&2 echo "Remediating rule 44/44: 'xccdf_org.ssgproject.content_rule_sshd_disable_root_login'")

# Remediation is applicable only in certain platforms
```

```
if [ ! -f /.dockerenv ] && [ ! -f /run/.containerenv ]; then


if [ -e "/etc/ssh/sshd_config" ] ; then


   LC_ALL=C sed -i "/^\s*PermitRootLogin\s\+/Id" "/etc/ssh/sshd_config"
else
   touch "/etc/ssh/sshd_config"
fi
# make sure file has newline at the end
sed -i -e '$a\' "/etc/ssh/sshd_config"


cp "/etc/ssh/sshd_config" "/etc/ssh/sshd_config.bak"
# Insert before the line matching the regex '^Match'.
line_number="$(LC_ALL=C grep -n "^Match" "/etc/ssh/sshd_config.bak" | LC_ALL=C sed
's/:.*//g')"
if [ -z "$line_number" ]; then
   # There was no match of '^Match', insert at
   # the end of the file.
   printf '%s\n' "PermitRootLogin no" >> "/etc/ssh/sshd_config"
else
   head -n "$(( line_number - 1 ))" "/etc/ssh/sshd_config.bak" > "/etc/ssh/sshd_config"
   printf '%s\n' "PermitRootLogin no" >> "/etc/ssh/sshd_config"
   tail -n "+$(( line_number ))" "/etc/ssh/sshd_config.bak" >> "/etc/ssh/sshd_config"
fi
# Clean up after ourselves.
rm "/etc/ssh/sshd_config.bak"


else
```

```
    >&2 echo 'Remediation is not applicable, nothing was done'
fi

# END fix for 'xccdf_org.ssgproject.content_rule_sshd_disable_root_login'
```

script 2:

https://github.com/konstruktoid/hardening

volg de stappen

manuele commando's:

```
sudo chmod 600 /boot/grub/grub.cfg
sudo apt install systemd-journal-remote
sudo apt-get remove iptables-persistent
sudo apt-get remove nftables
sudo systemctl enable ufw.service
sudo apt-get install ufw
sudo systemctl enable ufw.service
sudo sysctl -w fs.suid_dumpable=0

cd /etc/sysctl.d
fs.suid_dumpable = 0

sudo ufw allow 53
sudo ufw allow OpenSSH
sudo ufw reload

sudo chmod 0700 /var/log/audit
sudo nano /etc/motd
```