Automate Import and Export of Alias Configuration

Contents

Introduction

This document describes the steps to automate the tasks of Import and Export alias configuration within Email Security Appliance.

Prerequisites

Requirements

Cisco recommends knowledge of these topics:

- Cisco Secure Email Gateway (SEG / ESA) AsyncOS 16.0.2
- Command line interface access to Cloud Appliance
- Linux CLI
- Shell scripting

Components Used

The information in this document is based on these software:

- Cloud Email Security Appliance (CESA)
- Bash

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

The objective is to automate certain tasks, but some processes typically require manual intervention. However, in the case of importing and exporting the current alias configuration, these tasks can be fully automated, eliminating the need for manual input.

Exporting and Importing Alias Table

To import an alias table, start by verifying SSH, and SCP access to ensure you can connect to the email gateway.

Before proceeding, an alias table must exist within the appliance:

```
(Machine esa1.xyz.iphmx.com) (SERVICE)> clustermode cluster; aliasconfig print

test: test@example.com, test@example2.com, test@example3.com
test2: test@domain.com, test@domain2.com, test@domain3.com
(Cluster Hosted_Cluster) (SERVICE)>
```

When you use the **export** subcommand of the **aliasconfig** command to back up any existing alias table, a file (with a name you specify) is generated and saved in the /configuration directory for the listener.

Exporting Alias Table Using Bash Script

In this case, a bash script connects to your CES appliance and proceeds with exporting the alias file

The bash script is structured as shown:

```
#!/bin/bash

# Configuration of SSH keys and paths
PROXY_KEY="/full/path/folder/.ssh/id_rsa"
SECOND_KEY="/full/path/folder/.ssh/id_rsa"
LOCAL_PORT="2200"
PROXY_USER="dh-user"
PROXY_HOST="f4-ssh.iphmx.com"
TARGET_HOST="esa1.xyz.iphmx.com"
REMOTE_USER="local_server_user"
REMOTE_FILE="/configuration/filename.csv"
LOCAL_DIR="/full/path/folder/Downloads"
LOCAL_FILE_PATH="${LOCAL_DIR}/aliasconfig-file.csv"

# 1. Connect to the proxy and set up the SSH tunnel
echo "Establishing connection to the proxy..."
ssh -i "$PROXY_KEY" -l "$PROXY_USER" -N -f "$PROXY_HOST" -L "$LOCAL_PORT:${TARGET_HOST}:22"
if [ $? -ne 0 ]; then
    echo "Error: Failed to establish connection to the proxy."
    exit 1
fi
echo "Proxy connection established."

# Pause for 5 seconds before proceeding
sleep 5

# 2. Export the aliasconfig file from the remote system
echo "Exporting aliasconfig file from the remote system..."
ssh -i "$SECOND_KEY" "$REMOTE_USER"@127.0.0.1 -p "$LOCAL_PORT" 'clustermode cluster; aliasconfig export
if [ $? -ne 0 ]; then
    echo "Error: Failed to export the aliasconfig file."
    exit 1
fi
echo "Aliasconfig file successfully exported."

# Pause for 5 seconds before proceeding
sleep 5

# 3. Download the file to the local directory
echo "Downloading file to the local directory..."
scp -i "$SECOND_KEY" -P "$LOCAL_PORT" -O "$REMOTE_USER"@127.0.0.1:"$REMOTE_FILE" "$LOCAL_DIR" 2>/dev/nu
if [ $? -ne 0 ]; then
    echo "Error: Failed to download the file to the local directory."
    exit 1
fi
echo "File successfully downloaded to: $LOCAL_FILE_PATH"

# Pause for 5 seconds before finalizing
sleep 5

# Finalizing the script
echo "Process completed successfully."
exit 0
```

Explanation

1.- Configuration of SSH Keys and Paths

- PROXY_KEY and SECOND_KEY: The full path to the SSH private key file used for authentication. Both keys are set to the same path in this case.
- Example: /full/path/folder/.ssh/id_rsa
- LOCAL_PORT: Specifies the local port (2200) for the SSH tunnel.
- PROXY_USER: The username used for connecting to the proxy server.
- PROXY_HOST: The hostname of the proxy server.
- TARGET_HOST: The fully qualified domain name (FQDN) of the target host, updated to esa1.xyz.iphmx.com.
- REMOTE_USER: The username used for connecting to the remote appliance via the SSH tunnel.
- REMOTE_FILE: The path on the remote system where the exported file is stored (/configuration/filename.csv).
- LOCAL_DIR: The local directory for saving the file is set to /full/path/folder/Downloads.
- LOCAL_FILE_PATH: The full local path for the downloaded file, updated to ${LOCAL_DIR}/aliasconfig-file.csv.

2.- Connect to the Proxy and Set Up the SSH Tunnel

```
echo "Establishing connection to the proxy..."
ssh -i "$PROXY_KEY" -l "$PROXY_USER" -N -f "$PROXY_HOST" -L "$LOCAL_PORT:${TARGET_HOST}:22"
```

- Purpose:
  - Sets up an SSH tunnel to the proxy server for secure communication with the target host.
- Updates:
  - The SSH private key is now located at /full/path/folder/.ssh/id_rsa.
  - The target hostname has been updated to esa1.xyz.iphmx.com.
- Error handling:
  - f the connection fails, an error message is displayed, and the script exits with an error code (exit 1).

3.- Pause for 5 Seconds before Proceeding

- Purpose:
  - Introduces a delay to ensure the SSH tunnel is fully established before proceeding to the next step.

4.- Export the aliasconfig File from the Remote System

```
echo "Exporting aliasconfig file from the remote system..."
ssh -i "$SECOND_KEY" "$REMOTE_USER"@127.0.0.1 -p "$LOCAL_PORT" 'clustermode cluster; aliasconfig export
```

- Purpose:
  - Connects to the target host through the SSH tunnel and exports the alias configuration to a file named aliasconfig-file.csv.
- Updates:
  - The filename for the exported file has been updated to aliasconfig-file.csv.
- Output Redirection:
  - 2>/dev/null suppresses any error messages from the SSH command.
- Error Handling:
  - If the export fails, an error message is displayed, and the script exits.

5.- Download the File to the Local Directory

```
echo "Downloading file to the local directory..."
scp -i "$SECOND_KEY" -P "$LOCAL_PORT" -O "$REMOTE_USER"@127.0.0.1:"$REMOTE_FILE" "$LOCAL_DIR" 2>/dev/nu
```

- Purpose:
  - Uses scp to securely copy the exported file from the remote system to the local directory.
- Updates:
  - The local file is saved as aliasconfig-file.csv in the /full/path/folder/Downloads directory.
- Error Handling:
  - If the file download fails, an error message is displayed, and the script exits.

Finalizing the Script

```
echo "Process completed successfully."
exit 0
```

- Purpose:
  - Outputs a success message and exits the script with a success code (exit 0), indicating that all operations were completed successfully.

SSH Keys

You can note in the script two variables PROXY_KEY and HOST_KEY. These keys could be the same or different.

The PROXY_KEY is used to connect to the proxy cloud that is mandatory to jump to your CESA servers.

The HOST_KEY is the key used to log in as a local user, eliminating the need for a password.

Note: To configure SSH access to the CES proxy and set up an SSH key for a local user on the appliance, please consult the configuration guide.

Verifying the Exported File

Once the export script is executed, you can verify the content of it and you can observe that it contains the same information as the original presented within the appliance **aliasconfig** CLI command.

```
$ pwd
/full/path/folder/Downloads
$ ls
filename.csv
$ cat filename.csv
# File exported by the CLI at 20250702T125347
test: test@example.com, test@example2.com, test@example3.com
test2: test@domain.com, test@domain2.com, test@domain3.com⏎
```

Importing Alias Table Using Bash Script

Once you have exported the current alias file, you can modify it, add the necessary entries and then, import it to the ESA alia configuration.

The import bash script is structured as shown:

```
#!/bin/bash

# Configuration of SSH keys and paths
SSH_KEY="/full/path/folder/.ssh/id_rsa"
LOCAL_PORT="2200"
REMOTE_USER="local_server_user"
LOCAL_FILE="/full/path/folder/Downloads/new-filename.csv"
OUTPUT_DIR="/full/path/folder/Downloads"

# Get the current local date in the desired format
CURRENT_DATE=$(date +"%Y-%m-%d_%H-%M-%S")

# 1. Upload the new aliasconfig file
echo "Uploading new aliasconfig file to the remote system..."
scp -i "$SSH_KEY" -P "$LOCAL_PORT" -O "$LOCAL_FILE" "$REMOTE_USER"@127.0.0.1:/configuration 2>/dev/null
if [ $? -ne 0 ]; then
    echo "Error: Failed to upload the aliasconfig file."
    exit 1
fi
echo "Aliasconfig file successfully uploaded."

# Pause for 5 seconds before proceeding
sleep 5

# 2. Import the new aliasconfig file and commit with a comment
COMMIT_COMMENT="Importing new entries to aliasconfig - $CURRENT_DATE"
echo "Importing the new aliasconfig file and committing changes..."
ssh -i "$SSH_KEY" "$REMOTE_USER"@127.0.0.1 -p "$LOCAL_PORT" "clustermode cluster; aliasconfig import ne
if [ $? -ne 0 ]; then
    echo "Error: Failed to import the aliasconfig file or commit changes."
    exit 1
fi
echo "Aliasconfig file successfully imported and committed with comment: '$COMMIT_COMMENT'."

# Pause for 5 seconds before proceeding
sleep 5

# 3. Print the current aliasconfig and save it to a new file
OUTPUT_FILE="${OUTPUT_DIR}/current-aliasconfig-${CURRENT_DATE}.txt"
echo "Printing current aliasconfig and saving it to: $OUTPUT_FILE..."
ssh -i "$SSH_KEY" "$REMOTE_USER"@127.0.0.1 -p "$LOCAL_PORT" 'clustermode cluster; aliasconfig print' >
if [ $? -ne 0 ]; then
    echo "Error: Failed to print the current aliasconfig."
    exit 1
fi
echo "Current aliasconfig successfully saved to: $OUTPUT_FILE"

# Finalizing the script
echo "Process completed successfully."
exit 0
```

Explanation

1.- SSH Path and Key Configuration

- SSH_KEY: Path to the SSH private key file, used for securely authenticating against the remote server.
- LOCAL_PORT: Local port designated for the SSH tunnel.
- REMOTE_USER: User account for authentication on the remote server.
- LOCAL_FILE: Local path to the aliasconfig CSV file to be imported.
- OUTPUT_DIR: Local folder where a copy of the current configuration gets saved after the import process.

2.- Get Current Date and Time

```
CURRENT_DATE=$(date +"%Y-%m-%d_%H-%M-%S")
```

- Purpose:
  - Store the current date and time in a specific format for use in file names and comments.
- Updates:
  - Enables easy tracking and organization of logs and backups by timestamp.

3.- Upload the New aliasconfig File to the Remote ESA Server

This is the new content of the aliasconfig file:

```
# File exported by the CLI at 20250709T112719
test: new-data@example.com, new-date@example2.com, new-date@example3.com
test2: new-date@domain.com, new-data@domain2.com, new-data@domain3.com⏎
```

Proceed with the instruction to upload the file:

```
echo "Uploading new aliasconfig file to the remote system..."
scp -i "$SSH_KEY" -P "$LOCAL_PORT" -O "$LOCAL_FILE" "$REMOTE_USER"@127.0.0.1:/configuration 2>/dev/null
if [ $? -ne 0 ]; then
    echo "Error: Failed to upload the aliasconfig file."
    exit 1
fi
echo "Aliasconfig file successfully uploaded."
```

- Purpose:
  - Transfer the aliasconfig CSV file from the local machine to the /configuration directory on the remote server using SCP over SSH.
- Updates:

- If the upload fails, the script displays an error and stops to prevent incomplete imports.

4.- Import the New aliasconfig File and Commit with a Comment

```
COMMIT_COMMENT="Importing new entries to aliasconfig - $CURRENT_DATE"
echo "Importing the new aliasconfig file and committing changes..."
ssh -i "$SSH_KEY" "$REMOTE_USER"@127.0.0.1 -p "$LOCAL_PORT" "clustermode cluster; aliasconfig import ne
if [ $? -ne 0 ]; then
    echo "Error: Failed to import the aliasconfig file or commit changes."
    exit 1
fi
echo "Aliasconfig file successfully imported and committed with comment: '$COMMIT_COMMENT'."
```

- Purpose:
  - Connect via SSH, import the uploaded CSV file into the alias configuration, and commit the changes with a timestamped comment for tracking.
- Updates:
  - If the import or commit fails, an error message is shown and the script exits to keep configuration consistent.

5.- Print the Current aliasconfig and Save it to a New Local File

```
OUTPUT_FILE="${OUTPUT_DIR}/current-aliasconfig-${CURRENT_DATE}.txt"
echo "Printing current aliasconfig and saving it to: $OUTPUT_FILE..."
ssh -i "$SSH_KEY" "$REMOTE_USER"@127.0.0.1 -p "$LOCAL_PORT" 'clustermode cluster; aliasconfig print' >
if [ $? -ne 0 ]; then
    echo "Error: Failed to print the current aliasconfig."
    exit 1
fi
echo "Current aliasconfig successfully saved to: $OUTPUT_FILE"
```

- • Purpose:
  - Connect via SSH, print the current alias configuration, and save the output to a local, timestamped file for backup and auditing.
- Updates:
  - If the operation fails, the script displays an error and stops to avoid giving incomplete results.

Verifying Changes

Once the script is executed, you can verify the changes in the alias config table and check the commits in the system logs.

Verify New aliasconfig Table Entries

New changes have been applied to the table.

```
(Machine esa1.xyz.iphmx.com) (SERVICE)> clustermode cluster; aliasconfig print
```

```
test: new-data@example.com, new-data@example2.com, new-data@example3.com
test2: new-data@domain.com, new-data@domain2.com, new-data@domain3.com
```

Verify Commit Changes

This command enables you to track and review the changes committed to the ESA, including the user who made the change and the date it occurred.

```
(Machine esa1.xyz-66.iphmx.com) (SERVICE)> grep "commit" system_logs
Wed Jul  9 11:29:42 2025 Info: PID 95790: User local_server_user commit changes: Importing new entries
```

Script Flexibility

While this script is currently written in Bash, it can be easily adapted or rewritten in other scripting or programming languages-such as Python, PowerShell, or Perl-to better align with the preferences or requirements of different administrators and environments. This flexibility ensures that the core logic and workflow can be maintained while leveraging the language or tools that align most closely with your operational requirements.

Final Thoughts

This import/export script provides a practical and efficient solution for managing alias configurations directly on the appliance. By automating the upload, import, and backup of configuration files, administrators can safely and reliably introduce changes without manual intervention. The script not only streamlines the process but also ensures traceability through timestamped backups and commit comments.

Additionally, having such a script helps maintain consistency and compliance in your environment, especially when multiple changes or bulk updates are needed. Regular backups of the current configuration offer an added layer of security, enabling quick recovery or rollback if needed.

Overall, this approach empowers teams to manage configuration updates with greater confidence, control, and efficiency. For any future adjustments, the script can be easily adapted to handle other types of configuration files or further automate additional maintenance tasks as required.

Reference Links

- Accessing the Command Line Interface (CLI) of Your Cloud Email Security (CES) Solution
- CLI Instructions: PuTTY [Windows/PC Users]
- How to configure SSH Public Key Authentication for login to the ESA without a password