Check Point object translation script

June 1st 2020

Jordan Schraa

Executive Summary

These python scripts where created to help a customer clean up their rule base ahead of their migration to the cloud. The purpose of these scripts is to go over the rule base and translate IPs according regex_rules.csv file. First migration.py will be run which translate of the current network objects matching the left-hand column in the csv file and will create new objects which correspond to the right-hand column of the csv file. These new objects will be added into the same place rules as the original objects. Then once the correct functionality with the new objects is ensured migration-cleanup.py will be run. This will clean up the old objects. This script can also be used to demo the power of using the API with Check Point.

For easier navigation of the document you can use the Navigation pane which highlights each section.

Deployment and Testing

This guide will provide commands and screenshots that use the Linux command line; however, the procedure will be similar on a Windows desktop as well.

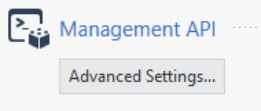
It is highly recommended to run the scripts against a backed-up copy of the management station and not a production server.

Dependences and Script Download

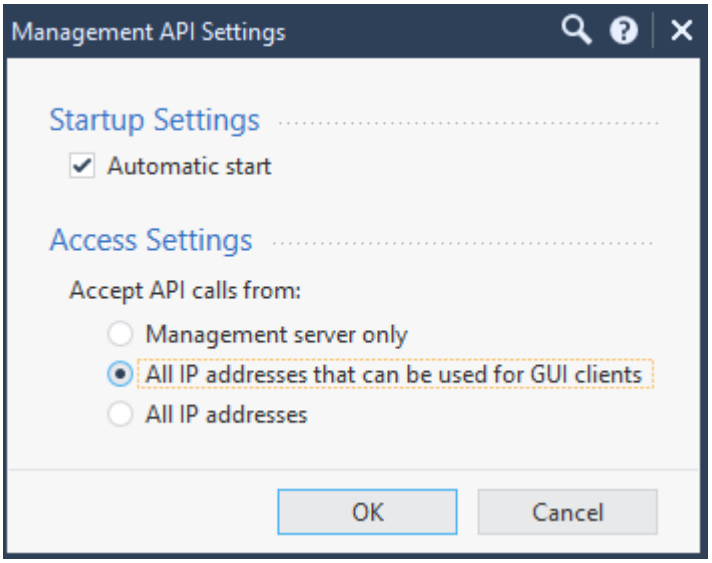
The first step is to download the dependencies and scripts to be able to run the commands.

The dependencies and download procedure are listed below:

- Python3
  - The download link for Python3 can be found here, however Python comes preinstalled on most Linux distributions: https://www.python.org/downloads/
- python requests
  - Once Python3 downloaded you can download the python requests library with the following command
    - **pip install requests**
- Check Point Management Server with API calls enabled. In the example I will be using a Management Server running R80.40 in Azure, but the script will work on any version of Check Point R80.10 and higher.
  - To check that the management API is up and running login to SmartConsole and navigate **Manage and Settings -> Blades -> Management API**



Make sure that the Access settings are set to either **All IP addresses that can be used for GUI clients** or **All IP addresses** since the script will be working off a separate server to send the commands to the management station. If you make changes to the management API settings remember to publish and restart the api with **api restart**



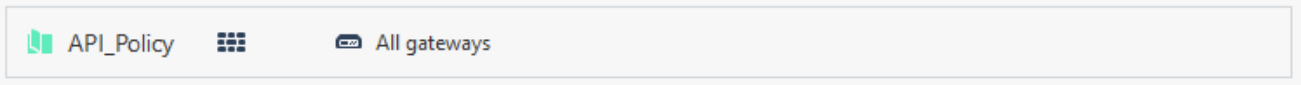
You can also verify API connectivity with the command **api status**

Once you have installed the dependencies you can download the files from GitHub. To download the files use **cd** to choose the directory you want to install the script folder into and the run the command **git clone https://github.com/jordanschraa/CheckPointRegexRules.git** This will install the scripts into that folder called CheckPointRegexRules.

Testing the Script

The github page includes scripts to test the functionality of the main migration.py and migration-cleanup.py scripts. To load the test data onto a management server run the command **python create_policy.py**

This script will connect to the management server and populate a new policy with some test objects and rules. You will see 200 code on the screen for every successful object that is created. The new policy that will be created is API_Policy. It should look like the policy below.



No.	Name	Source	Destination	VPN	Services & Applications	Content	Action	Track
1	Management_Rule	Admin	cpManagment	Any	Any	Any	Accept	Log
2	Stealth_Rule	Any	cpGateway	Any	Any	Any	Drop	Log
3	Remote_Access	RemoteUser	SQL_Server1 SQL_Server2	Any	Any	Any	Accept	Log
4	Web_Access	InternalUser1 InternalUser2	Internet	Any	Facebook Real Estate	Any	Drop	Detailed L Accou
5	Web_Server_Update	Web_Server1 Web_Server2 Web_Server3	Internet	Any	Windows Server Up...	Any	Accept	Log Accou
6	Cleanup rule	Any	Any	Any	Any	Any	Drop	None

The file regex_rules.csv will contain all the information for translating the objects. For the purposes of testing you can change these values to be anything. “*” denotes a wildcard and will match with any octet if it is in the original column, if the star is in the translated column it means it will not change that value when translating the IP addresses.

	A	B
1	original	translated
2	10.0.1.*	10.5.1.*
3	192.168.26.*	192.168.54.*
4	192.168.5.*	192.168.35.*
5	172.168.*.5	172.0.*.5

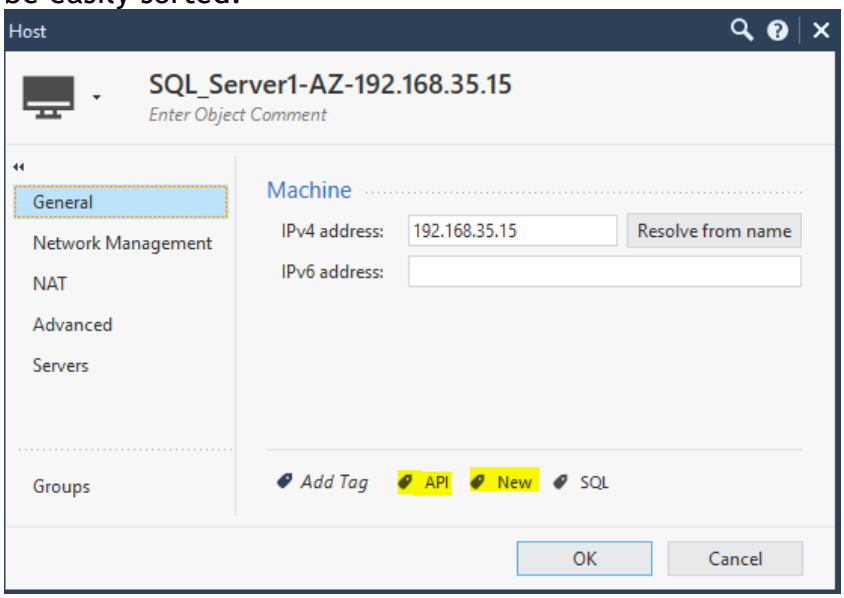
Next run the command **python migration.py** command and the script will iterate over the rulebase to make changes according to the regex_rules.csv The script will output the rules one at a time and list any changes being made. When the script is finished the session will be published in SmartConsole. The output of the script will look the terminal below.

```
λ python migration.py

Authenticating to Managment with following credentials:
Managment server IP 13.92.42.91
Username admin
Password ****************
Port 443

Enter 'y' to continue, 'n' to change values and 'z' to quit: y
Authentication Successful
      --- Management_Rule ---
      --- Stealth_Rule ---
      --- Remote_Access ---
Matched on rule #3
Orginal: 192.168.5.15 New: 192.168.35.15
Matched on rule #3
Orginal: 192.168.5.25 New: 192.168.35.25
      --- Web_Access ---
      --- Web_Server_Update ---
Matched on rule #1
Orginal: 10.0.1.15 New: 10.5.1.15
Matched on rule #1
Orginal: 10.0.1.25 New: 10.5.1.25
Matched on rule #1
Orginal: 10.0.1.35 New: 10.5.1.35
      --- Cleanup rule ---
```


The new objects created in Smart Console will look like this. *oldname+AZ+newIPaddress* this new name is used to easily differentiate new objects from old ones. The new objects are also created with the tag **API** and **New** so they can be easily sorted.



Next run the command **python migration-cleanup.py** this command will go through the policy and clean up the old objects and publish the policy afterwards.

```
λ python migration-cleanup.py

Authenticating to Managment with following credentials:
Managment server IP 13.92.42.91
Username admin
Password ****************
Port 443

Enter 'y' to continue, 'n' to change values and 'z' to quit: y
Authentication Successful
Removing SQL_Server1
Removing SQL_Server2
Removing Web_Server1
Removing Web_Server2
Removing Web_Server3
```

Now the new policy will only have the new migrated objects in the policy

No.	Name	Source	Destination	VPN	Services & Applications	Content	Action	Track
1	Management_Rule	Admin	cpManagment	Any	Any	Any	Accept	Log
2	Stealth_Rule	Any	cpGateway	Any	Any	Any	Drop	Log
3	Remote_Access	RemoteUser	SQL_Server1-AZ-192.168.3... SQL_Server2-AZ-192.168.3...	Any	Any	Any	Accept	Log
4	Web_Access	InternalUser1 InternalUser2	Internet	Any	Facebook Real Estate	Any	Drop	Detailed L Accou
5	Web_Server_Update	Web_Server1-AZ-10.5.1.15 Web_Server2-AZ-10.5.1.25 Web_Server3-AZ-10.5.1.35	Internet	Any	Windows Server Up...	Any	Accept	Log Accou
6	Cleanup rule	Any	Any	Any	Any	Any	Drop	None

Running the Script

The first step in running the script is to edit the regex_rules.csv file so that it is applicable to the environment the script is being run in. This is where the rules must be created, if they are created in a different folder or under a different name the contents will not be read by the script. After this file has been edited you can move onto the next section which is running the script.

When you are running the script for the first time make sure you are in the CheckPointRegexRules directory and run the command **python migration.py** This will guide you through the steps of connecting to your management server. The prompt should look something like below. **NOTE:** The credentials are stored in plain text in a file called credentials.json for ease of use. If you do not want the password stored in plain text remember to delete this file.

```
λ python migration.py
Enter managment server IP address: 192.168.1.1
Enter username: admin
Enter password:
Press enter for default port: 443
Enter port number:

Authenticating to Managment with following credentials:
Managment server IP 192.168.1.1
Username admin
Password ************
Port 443

Enter 'y' to continue, 'n' to change values and 'z' to quit:
```

After the **migration.py** script has been ran, verify that all the rules were edited as expected and then test the configuration. Once the configuration has been verified and is working the **migration-cleanup.py** file can be run to clean up the rule base.

The **migration-cleanup.py** functions just like it would when the script was ran in testing.

Additional information on the files included

header.py

- header.py can be used as the basis for python programs with Check Point API
- contains two usefull functions authenticate and api_post
    - authenticate displays collects Check Point management credentials from the user and writes them to credentials.json for easier logins in the future. Collection of credentials done with a menu style. Authenticate will also handle the first time login and will return dictonary that contians sessions id and logon credentials
    - api_post handles the api calls, you must provide the credentials that were returned by the authenticate method

- ex: creds = authenticate() r, c = api_post(creds, "add-host", json-host)

create_policy.py

- create_policy.py will use the data in objects.json and policy.json to create an example policy to demonstrate API capabilities

migration.py

- migration.py is a program to help migration of a customer to Azure
- it will go over a rule set one rule at time and find objects in the source and destination columns that match the regex rules that are defined in csv file
- if the object matches a regex rule it will have a tag of "Old" added to it this is for cleanup purposes later
- once it finds a match to the regex rule it will create a duplicate object that changes the IP based on the regex rule and adds "-NEW" to the name of the object. This new object will have a tag of "New" added to it
- then it inserts the new object into the same position in the ruleset
- all the changes will be visible at the end when the session is published

migration-cleanup.py

- migration-cleanup.py cleans up the rule set after migration.py has been run
- once you you have done the tests to determine that ruleset is functioning correctly after running migration.py you can run migration-cleanup.py to clean up the rule set again
- this program goes over the rule set one rule at a time and finds objects with the old tag and removes them from the policy.
- this object will not be deleted from the data base (may change this functionality later?)
- it will edit the newly created objects to remove the "-NEW" from the name and remove the "New" tag
- all the changes will be visible at the end when the session is published

Limitations

- Multi-Domain environments are not currently supported
- Group objects are not supported

To submit any issues or feature requests please enter them into the issues page on GitHub.
https://github.com/jordanschraa/CheckPointRegexRules/issues

