

PAN-OS CLI w/ Automation for NGFW and Panorama

Data collection and Visualization

Initiative of the piece of work

- Develop expertise for advanced POC's.
- Support unique PAN solutions for the field, with expertise of the team.
- PAN-OS feature showcase to enable customer demos and evaluations.
- Train and help understand new and unique features of our products.
- Proven setups for linked demos and sandboxes.
- Establishment for further proof of functions, performance and capacity.
- Documentation to promote proven, sized and repeatable setups.
- Last but not least, establishment for further development and possibilities.



Documentation as additional references

POC Team is consistently developing technical content to accelerate testing and POC cycles. Here this is the collection of documents for various use cases. We welcome your feedback to help enhance our productivity with detailed and efficient documentation.

https://drive.google.com/drive/folders/1azpLLToTYzfwynAF4gNImC1W9ViaL9Xq?usp=drive_link

Note 10 - PAN-OS CLI with Python

Note 9 - Subscriber-ID

Note 8 - PAN-OS API with Python

Note 7 - Explicit Web proxy w/ Network Packet Broker

Note 6 - PAN-OS SD-WAN

Note 5 - KVM w/ OVS-DPDK

Note 4 - Prisma SD-WAN POC setup v2.0

Note 3 - MFA

Note 2 - GlobalProtect w/ AD

Note 1 - User-ID

Developed tools on Github for POC and system admin <https://github.com/teleee0>

Revision history

20250616 - Initial release

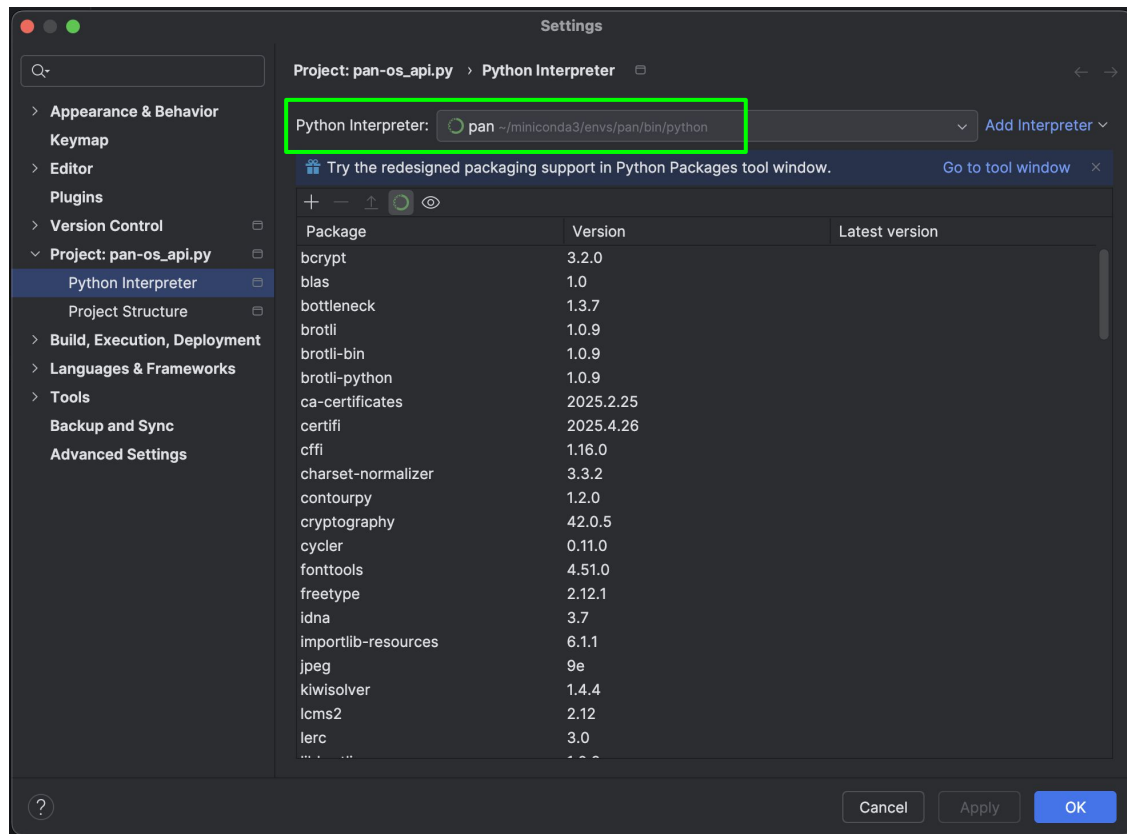
PAN-OS CLI with Python

- Tools shared through GitHub https://github.com/telee0/pan-os_cli
- Scripts have been rewritten in Python to support multiple-stage execution model. Each stage is modeled as a distinct state with finite iteration counts.

```
$ python3 pan-cli.py -c conf/cli.py -h <PA1_host> -u admin -v
```

- Collected device response as command output for downstream tasks, such as data analysis, summary statistics and visualization (e.g. DP CPU graphs).
- Tested on Windows, MacOS, and Linux (CentOS, Ubuntu and Ubuntu/WSL)
- Used preferably with IDE (PyCharm) so executions and configuration files (conf/*.py) can be managed easily.
- Access password may be specified in an environment variable, so that it will not be hardcoded in the configuration file.

Pycharm project settings



- Dedicated virtual runtime environment is recommended.
- This ensures a stable environment for the scripts from broken by package upgrade for other projects.
- Virtual runtime environment can be created with venv, miniconda, etc.
- You may check with chatGPT about these options for virtualized runtime environment.

Pycharm run configuration

- The main script is `pan-cli.py`

- Script parameters:

`-c conf/cli.py` config file

`-h 192.168.1.1` target host

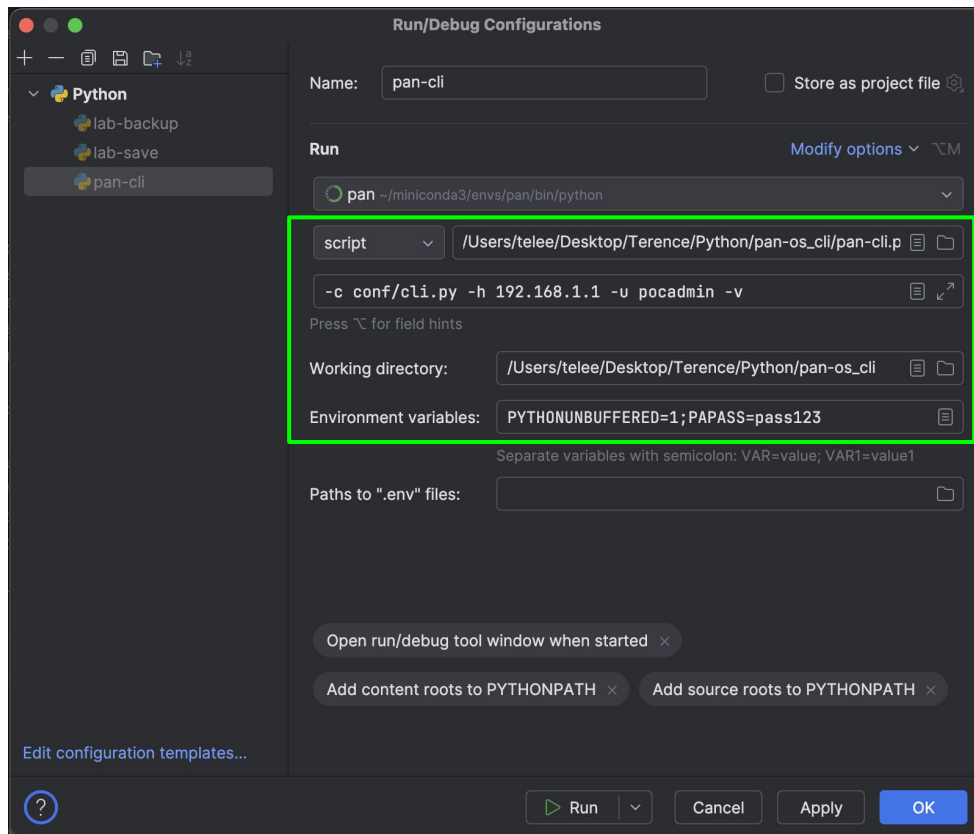
`-u admin` login user name

`-v` verbose mode for more details

- Environment variable for the password:

`PAPASS=pass123` login password

So you do not need to keep your passwords in the files before saving them on shared drives.



Required (Python) packages for the scripts

- There is a file `requirements.txt` which is recognized by PyCharm so can be used to patch the environment.
- The environment also can be checked for installed packages with `pip`.
- Use `pip install -r requirements.txt` to install the packages.

```
(pan) C:\Users\Terence>conda activate pan
(pan) C:\Users\Terence>pip list --format=freeze
Brotli==1.0.9
certifi==2025.4.26
charset-normalizer==3.3.2
idna==3.7
pip==25.1
PySocks==1.7.1
requests==2.32.3
setuptools==78.1.1
urllib3==2.3.0
wheel==0.45.1
win_inet_pton==1.1.0

(pan) C:\Users\Terence>
```

Win 11

```
(base) telee@M-KGVMGHP6FC ~ % conda activate pan
(pan) telee@M-KGVMGHP6FC ~ % pip list --format=freeze
bcrypt==3.2.0
Bottleneck==1.3.7
Brotli==1.0.9
certifi==2025.4.26
cffi==1.16.0
charset-normalizer==3.3.2
contourpy==1.2.0
cryptography==42.0.5
cyclerr==0.11.0
fonttools==4.51.0
idna==3.7
importlib-resources==6.1.1
kiwisolver==1.4.4
matplotlib==3.8.4
numexpr==2.8.7
numpy==1.26.4
packaging==23.2
pandas==2.2.1
paramiko==2.8.1
paramiko-expect==0.3.5
pillow==10.3.0
pip==24.0
pycparser==2.21
PyNaCl==1.5.0
pyparsing==3.0.9
PySocks==1.7.1
python-dateutil==2.9.0.post0
pytz==2024.1
requests==2.32.3
scipy==1.11.4
setuptools==69.5.1
```

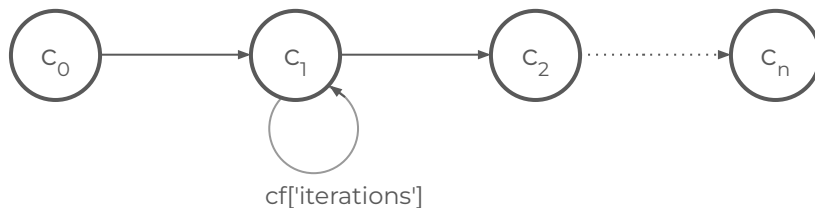
MacOS

```
six==1.16.0
tornado==6.3.3
tzdata==2023.3
unicodedata2==15.1.0
urllib3==2.3.0
wheel==0.43.0
zipp==3.17.0
(pan) telee@M-KGVMGHP6FC ~ %
```

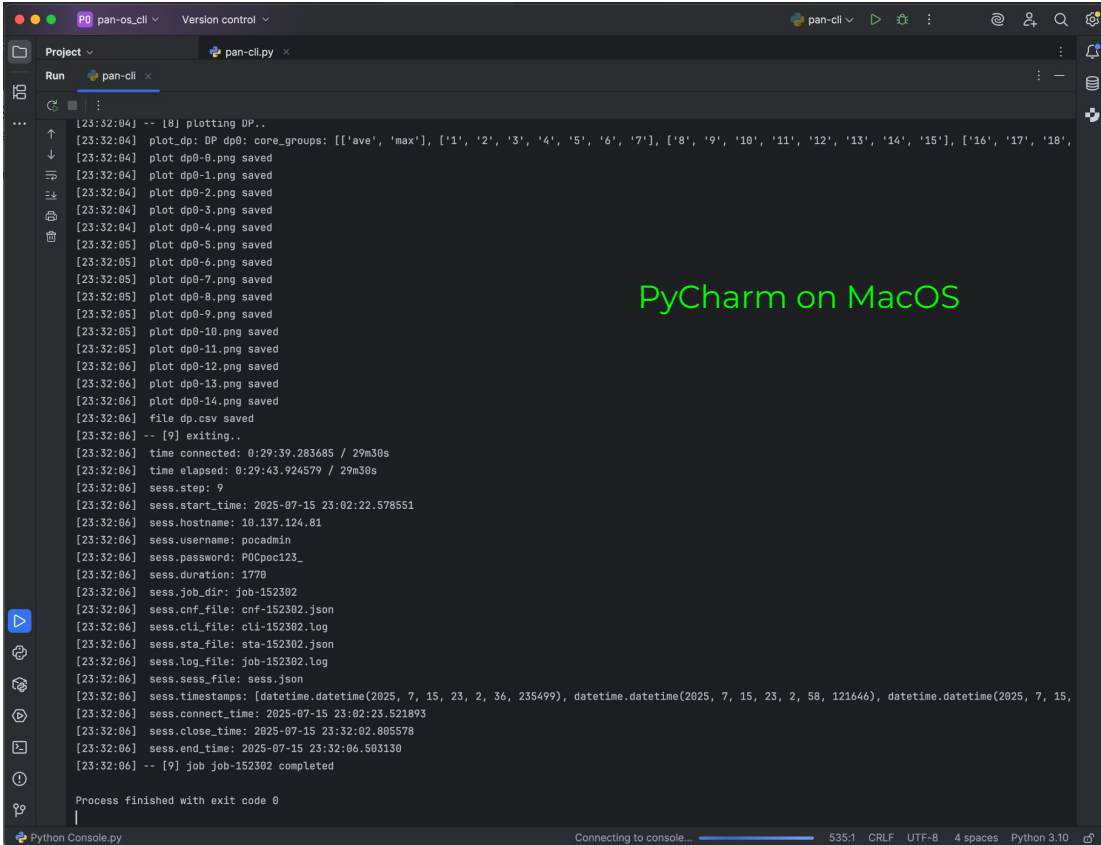

Multiple-state execution model

- Configuration as a python script defines different command sets as distinct states, each with the command set repeated k times (e.g. cf['iterations'] for c1) through SSH.

```
conf/cli.py:  
  
cli = [  
    c0,  
    (c1, cf['iterations']), # c1 may be different for POC's e.g. cli = [c0, c_poc12345, c2]  
    c2  
]
```



Sample output



The image shows a PyCharm IDE window on a Mac. The main editor area displays a file named `pan-cli.py`. The terminal window at the bottom shows the output of a Python script. The output includes a list of saved plot files (dp0-0.png to dp0-14.png), a CSV file (dp.csv), session information (start time, hostname, username, password, duration, job directory, etc.), and timestamps for various events. The text "PyCharm on MacOS" is overlaid in green on the right side of the terminal output.

```
[23:32:04] -- [8] plotting DP..  
[23:32:04] plot_dp: DP dp0: core_groups: [['ave', 'max'], ['1', '2', '3', '4', '5', '6', '7'], ['8', '9', '10', '11', '12', '13', '14', '15'], ['16', '17', '18',  
[23:32:04] plot dp0-0.png saved  
[23:32:04] plot dp0-1.png saved  
[23:32:04] plot dp0-2.png saved  
[23:32:04] plot dp0-3.png saved  
[23:32:04] plot dp0-4.png saved  
[23:32:05] plot dp0-5.png saved  
[23:32:05] plot dp0-6.png saved  
[23:32:05] plot dp0-7.png saved  
[23:32:05] plot dp0-8.png saved  
[23:32:05] plot dp0-9.png saved  
[23:32:05] plot dp0-10.png saved  
[23:32:05] plot dp0-11.png saved  
[23:32:06] plot dp0-12.png saved  
[23:32:06] plot dp0-13.png saved  
[23:32:06] plot dp0-14.png saved  
[23:32:06] file dp.csv saved  
[23:32:06] -- [9] exiting..  
[23:32:06] time connected: 0:29:39.283685 / 29m30s  
[23:32:06] time elapsed: 0:29:43.924579 / 29m30s  
[23:32:06] sess.step: 9  
[23:32:06] sess.start_time: 2025-07-15 23:02:22.578551  
[23:32:06] sess.hostname: 10.137.124.81  
[23:32:06] sess.username: pocadmin  
[23:32:06] sess.password: POCpoc123_  
[23:32:06] sess.duration: 1770  
[23:32:06] sess.job_dir: job-152302  
[23:32:06] sess.cnf_file: cnf-152302.json  
[23:32:06] sess.cli_file: cli-152302.log  
[23:32:06] sess.sta_file: sta-152302.json  
[23:32:06] sess.log_file: job-152302.log  
[23:32:06] sess.sess_file: sess.json  
[23:32:06] sess.timestamps: [datetime.datetime(2025, 7, 15, 23, 2, 36, 235499), datetime.datetime(2025, 7, 15, 23, 2, 58, 121646), datetime.datetime(2025, 7, 15,  
[23:32:06] sess.connect_time: 2025-07-15 23:02:23.521893  
[23:32:06] sess.close_time: 2025-07-15 23:32:02.805578  
[23:32:06] sess.end_time: 2025-07-15 23:32:06.503130  
[23:32:06] -- [9] job job-152302 completed  
  
Process finished with exit code 0
```

PyCharm on MacOS

Configuration files

- They are saved as Python files and can be specified through the command line.
- In order to perform data collection against multiple targets (e.g. HA), multiple runtime environments are required, such as multiple VM's, multiple Anaconda prompts, multiple PyCharm projects, etc.

Main configuration

```
cf = {
    'hostname': '192.168.1.1',      # host name or IP of the target device
    'username': 'admin',           # sensitive and not exported, default admin
    'password': '',                # sensitive and not exported, either here or through the env variable cf['passenv']
    'passenv': 'PAPASS',           # name of the environment variable for the password
    'prompt': r'.*>\s+',            # regex so prefixed with an 'r'

    # duration <= time_delay + time_intervals * (iterations - 1)

    'duration': '5m',              # max duration (? d ? h ? m ? s) <= time_delay + time_interval * (iterations - 1)
    'iterations': 999,             # max number of iterations
    'time_delay': 10,              # initial delay in seconds
    'time_interval': 20,           # interval in seconds between iterations

    'cli': 'cli_1',                # currently not used
    'cli_timeout': 5,              # > 0 or the cli will not expect a prompt

    'job_dir': 'job-{}',           # job folder
    'log_file': 'job-{}.log',       # job log
    'cnf_file': 'cnf-{}.json',      # config dump
    'cli_file': 'cli-{}.log',       # CLI output
    'sta_file': 'sta-{}.json',      # stats
    'sess_file': 'sess.json',       # session data

    'plot_file': 'p{0}-{1}.png',    # stats plot file names
    'plot_file_merged': 'p-{0}.png', # state merged plot file names

    'log_buf_size': 99,             # log buffer size in message count

    'verbose': True,
    'debug': False,
}
```

Configuration about device access

```
'hostname': '192.168.1.1',    # host name or IP of the target device
'username': 'admin',         # sensitive and not exported, default admin
'password': '',              # sensitive and not exported, either here or through the env variable cf['passenv']
'passenv': 'PAPASS',         # name of the environment variable for the password
'prompt': r'.*>\s+',         # regex so prefixed with an 'r'
```

- The first few config items should not be modified.
- Access details such as device IP, login name and password can be overridden with the command line options.

Configuration about command repetition

```
'duration': '5m',           # max duration (? d ? h ? m ? s) <= time_delay + time_interval * (iterations - 1)
'iterations': 999,         # max number of iterations
'time_delay': 10,          # initial delay in seconds
'time_interval': 20,       # interval in seconds between iterations
```

- Max duration of script execution is given by the configuration parameter 'duration'.
- Max number of iterations, 'iterations', can be adjusted if 'duration' can accommodate more iterations.
- 'time_delay' is introduced once, right after the target device is connected.
- 'time_interval' is the time between iterations of execution of the command set.
- This summarizes the relationship of the configuration parameters.

Execution time = min { duration, [time_delay + time_interval * (iterations - 1)] }

Configuration about output files

```
'job_dir': 'job-{}',      # job folder
'log_file': 'job-{}.log',  # job log
'cnf_file': 'cnf-{}.json', # config dump
'cli_file': 'cli-{}.log',  # CLI output
'sta_file': 'sta-{}.json', # stats
'sess_file': 'sess.json',  # session data

'plot_file': 'p{0}-{1}.png',      # stats plot file names
'plot_file_merged': 'p-{0}.png',  # state merged plot file names
```

- Path names are suffixed by `ddhhmm = datetime.now().strftime('%d%H%M')`
- `'cli_file'` captures device response.
- `'sta_file'` saves metric values specified in the dictionary **metrics** in the same file, which will be covered later.
- `'plot_file'` specifies the file name pattern of plot files (metrics).
- `'plot_file_merged'` is the file name pattern of multiple plot files put together in a grid.

Configuration about logging and debugging

```
'log_buf_size': 99,          # log buffer size in message count  
  
'verbose': True,  
'debug': False,
```

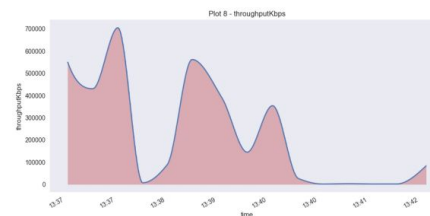
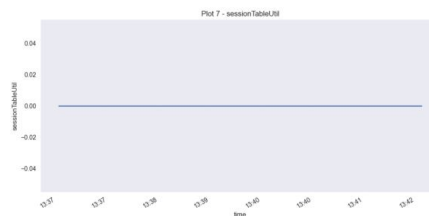
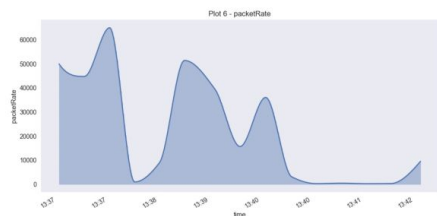
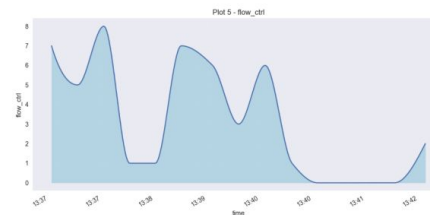
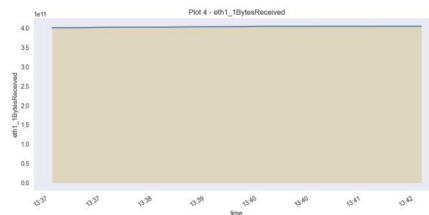
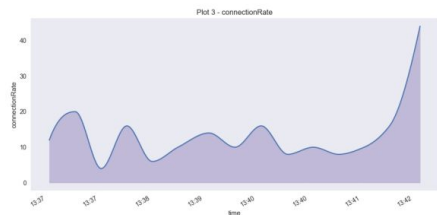
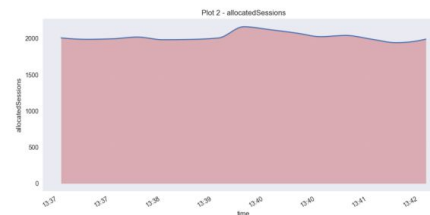
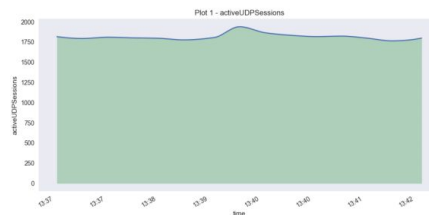
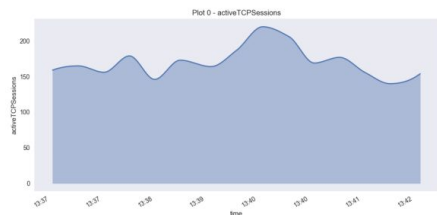
- These are pretty self explanatory.

Configuration about metric extraction

```
metrics = {
    'activeTCPSessions':    r'active TCP sessions:\s+(\d+)',
    'activeUDPSessions':    r'active UDP sessions:\s+(\d+)',
    'allocatedSessions':    r'allocated sessions:\s+(\d+)',
    'connectionRate':       r'connection establish rate:\s+(\d+) cps',
    'eth1_1BytesReceived':  r'bytes received\s+(\d+)',
    # 'eth1_43BytesReceived': r'bytes received\s+(\d+)',
    'flow_ctrl':            r'flow_ctrl\s+:\s+(\d+)%',
    'packetRate':           r'Packet rate:\s+(\d+)\s/s',
    'sessionTableUtil':     r'Session table utilization:\s+(\d+)%',
    'throughputKbps':       r'Throughput:\s+(\d+) kbps',
    # 'vpnIPSecTunnels':     r'Total (\d+) tunnels found',
}
```

- All these are regex, patterns for how metric values are extracted from device responses.
- For example, we may want to capture the concurrent sessions from device's perspective over the execution period. We have 'allocatedSessions' uncommented to capture it.
- The CLI command to check for this metric is found in c2, which is usually `show session info`.
- Over iterations, there are multiple responses (command output) collected. For each of these responses, the pattern is used to match the interesting line, so that the value for concurrent sessions can be eventually extracted.
- There are some metric patterns not applicable to all models. For example, 'eth1_43BytesReceived' is supposed for bytes received by eth1/43 (with a corresponding command in c2) . However not all targets have eth1/43.

Sample graphs generated for the metrics (grid 3x3)

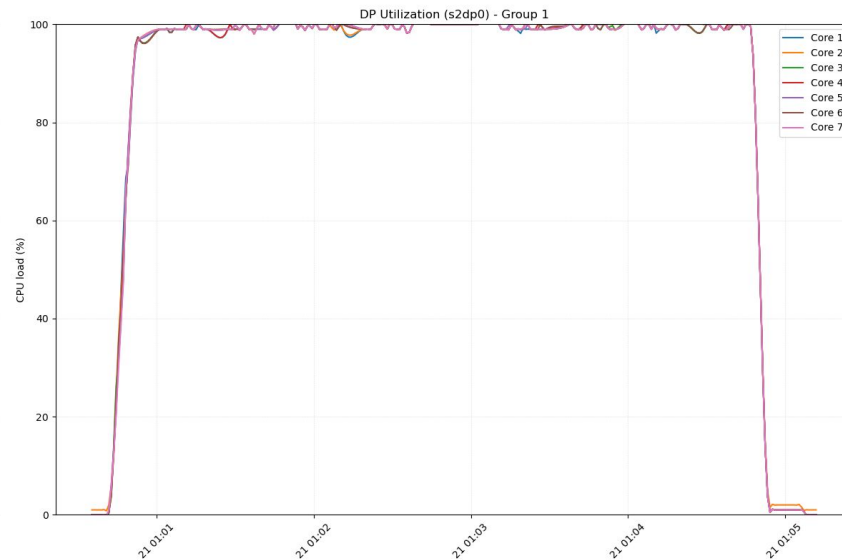
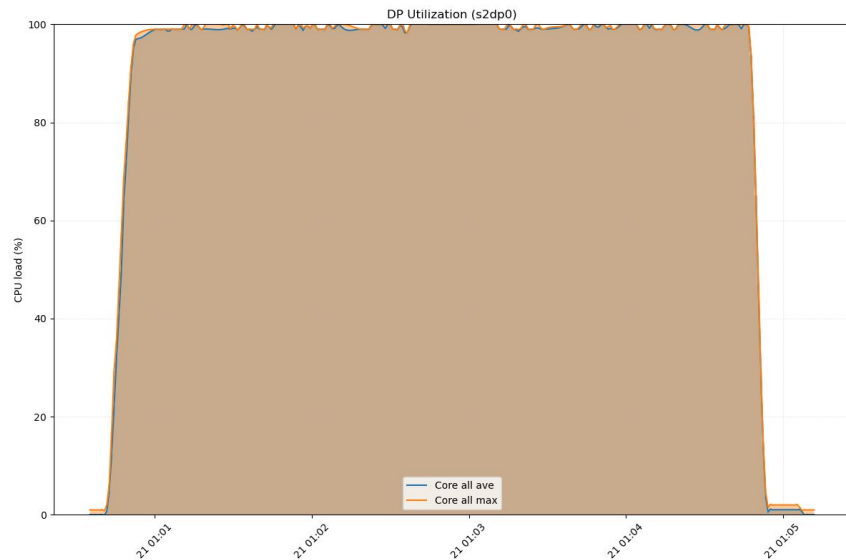


Configuration specific to dataplane (DP) visualization

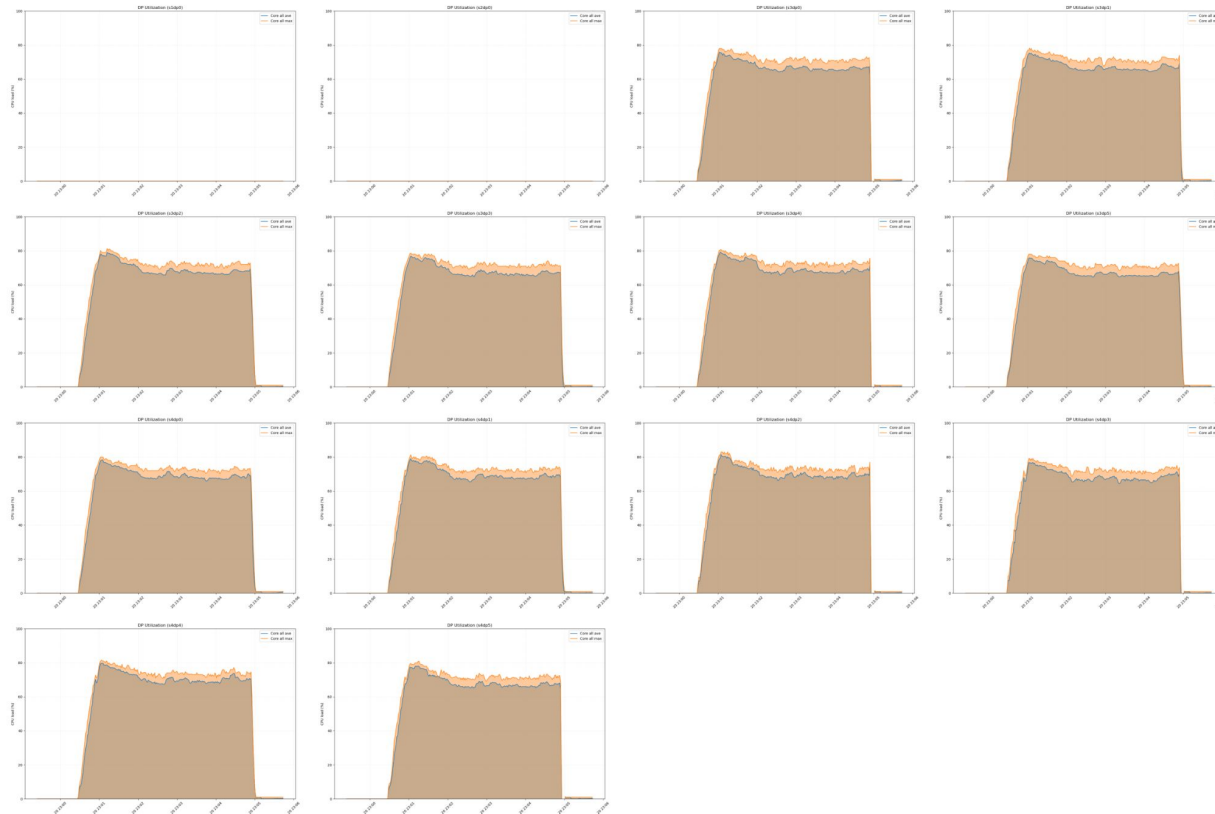
```
dp = {
    'command':          r'show\s+running\s+resource-monitor',
    'dp_name':          r'^DP\s+(s\d+dp\d+):',
    'cpu_load':         r'CPU load \((%\)\) during last (\d+) seconds:',
    'core':             'core',
    'dp_name_default':  'dp0', # do not use dp use dp0
    'cores_per_group':  8,
    'aggregate':        ('ave', 'max'), # , 'min'), # , 'sum', 'cnt'),
    'csv_file':         "dp.csv",
    'json_file':        "dp.json",
    'plot_file':        "{0}-{1}.png",
    'plot_file_merged': "dp-{0}.png",
    'plot_grid_size':   (3, 3), # (rows, columns) for merged summary plots
    'skip_first_row':   True, # address the issue where the most recent data row is incomplete (all low values)
    'skip_dp_names':    ["s1dp0", "s2dp0"], # currently not used
}
```

- Most of the DP config should remain default.
- There are some config parameters which can still be customized.
- 'cores_per_group' visualize core utilization in groups of 8.
- 'plot_file' specifies the file name pattern of plot files (DP cores not metrics previously seen).
- 'plot_file_merged' is the file name pattern of multiple plot files put together in a grid.
- 'plot_grid_size' defines the grid in which the plot files are organized.
- 'skip_first_row' is a workaround parameter for the issue where the most recent data row from `show running resource-monitor` is incomplete.

Sample graphs generated for DP utilization (all cores and a core group of 8)



Sample output for POV delivery (grid 4x4)



Sample configuration: c0

Command set 0 - c0

```
c0 = [  
    'show clock',  
    'set cli pager off',  
    'show system info',  
    'show chassis power',  
]
```

- C0 is specified to not repeat
- There are sample commands which are applicable to a subset of PA models. For example, `show chassis power` is applicable only to PA-7000 series, PA-5450 and PA-7500. The target device may respond saying the command line is not recognized, or “Invalid syntax”. This is not important but can be commented out to just save execution time.

Sample configuration: c1

Command set 1 - c1

```
c1 = [  
    'show session info',  
    'set cli pager on',  
    'show session all',  
    (' ', 2, 0), # space for the next page, totally 3 pages  
    ('q', 1, 0), # q to stop  
    'set cli pager off',  
    'show running resource-monitor second last 30',  
    'show system resources | match ": "',  
    # 'debug dataplane show ssl-decrypt ssl-stats',  
    # ('show session all filter count yes ssl-decrypt yes', 1, 10), # to trace decrypted sessions  
    # ('show session all filter count yes ssl-decrypt no', 1, 10), #  
    # 'show counter global | match proxy_process', # target global counters  
    # 'show counter global | match session_installed', #  
    # 'show counter global filter delta yes severity warn',  
    'show interface ethernet1/1 | match "bytes received"',  
    # 'show interface ethernet1/43 | match "bytes received"',  
    'show session distribution policy', # chassis models  
    # 'show interface all | match ^node', # pa-7500 cluster  
    # 'show cluster nodes', # pa-7500 cluster  
    # 'show vpn ipsec-sa summary | match "tunnels found"',  
    # 'show global-protect-gateway statistics',  
    # 'show lockless-qos enable',  
    # 'show lockless-qos if-core-mapping',  
    ('show clock',),  
]
```

Command set 1 - c1

- C1 is specified to repeat regularly for the most common use case.
- The command set is executed in sequential order of the commands
- This subset looks a bit weird, but is interpreted as follows.

```
'set cli pager on',  
'show session all',  
(' ', 2, 0), # space for the next page, totally 3 pages  
('q', 1, 0), # q to stop  
'set cli pager off',
```

'set cli pager on' turns on pager, so long output can be returned page by page. With pager turned on, we can control when to stop from 'show session all'. (' ', 2, 0) is the next command with a tuple structure, which is interpreted as pressing ' ' (space) twice (2) and do not wait for the prompt (0 means do not wait for it). Similarly the next one ('q', 1, 0) means pressing 'q' to stop and then 'set cli pager off' turns the pager off.

- There are sample commands which are applicable to a subset of PA models. For example, 'show session distribution policy' is applicable only to chassis models which consist of multiple DP cards. The target device may respond saying the command line is not recognized, or "Invalid syntax". This is not important but can be commented out to just save time.

Demo



Documentation as additional references

POC Team is consistently developing technical content to accelerate testing and POC cycles. Here this is the collection of documents for various use cases. We welcome your feedback to help enhance our productivity with detailed and efficient documentation.

https://drive.google.com/drive/folders/1azpLLToTYzfwynAF4gNImC1W9ViaL9Xq?usp=drive_link

Note 10 - PAN-OS CLI with Python

Note 9 - Subscriber-ID

Note 8 - PAN-OS API with Python

Note 7 - Explicit Web proxy w/ Network Packet Broker

Note 6 - PAN-OS SD-WAN

Note 5 - KVM w/ OVS-DPDK

Note 4 - Prisma SD-WAN POC setup v2.0

Note 3 - MFA

Note 2 - GlobalProtect w/ AD

Note 1 - User-ID

Developed tools on Github for POC and system admin <https://github.com/teleee0>

Thank you

