**DefensePro – Best Practice configuration analyzer:**

DefensePro Best Practice Configuration Recommendations report

Contents

[ChangeLog 3](#_Toc84418436)

[Overview 4](#_Toc84418437)

[Script Output 5](#_Toc84418438)

[DefensePro Best Practice Configuration Recommendations report (dpconfig\_report.csv) 5](#_Toc84418439)

[Setup 6](#_Toc84418440)

[Requirements 6](#_Toc84418441)

[Libraries/packages in use 6](#_Toc84418442)

[Instructions and recommendations 6](#_Toc84418443)

[Components for the script operation 7](#_Toc84418444)

[“config.py” 7](#_Toc84418445)

[“main.py” 7](#_Toc84418446)

[“vision.py” 8](#_Toc84418447)

[logging\_helper.py 8](#_Toc84418448)

[dpconfig\_parser.py 8](#_Toc84418449)

[Script operation stages 8](#_Toc84418450)

[Stage 1- Data collection 8](#_Toc84418451)

[Files necessary for the Best practice configuration review 8](#_Toc84418452)

[Stage 2- Data parsing 9](#_Toc84418453)

[Stage 3 – email alert 9](#_Toc84418454)

# ChangeLog

v1.3

- Added running dpconfig data parsing with cache data

--use-cache-data

v1.4

- dpconfig\_parser.py

- Change log to csv instead of dictionary

- Code revise

v1.5

- dpconfig\_parser.py

- If no profiles, no need to log missing specific profiles

- Fixed ERT Active Attacker Feed profile logging

- Added BDOS profile is in "Report-Only" mode

v1.6

- dpconfig\_parser.py

- Fixed bug with policies priorities in ver 7.x while policy name/policy priority is 'null'

V1.9

* Code optimization

V1.13

* Code optimization, added folders Requests, Raw Data, Reports

V2.0

* Split config analyzer as a dedicated app

V2.1

* Check BDOS profile Footprint Strictness and alert if not set to Medium
* Added printing and logging of script progress

Functionality to be added:

# Add profile state block/report

# Add recommendation priority High/Med/Low

# Add network class parsing

# Add config parsing additional tests

# Overview

The solution was developed through collaborated effort of the Radware team, initiated by Lior Rozen and the team - Marcelo Dantas, Daniel Offek, Egor Egorov, Sagiv Oron, Artur Mehitrian) in 2021.

The script purpose is to provide DefensePro Best Practice Configuration Recommendations report (dpconfig\_report.csv)

The script interacts with Radware APSolute Vision DefensePro and collects all the necessary data through REST API calls.

IMPORTANT● Read the entire file before attempting to configure/executing.

# Script Output

The script output generates one report in csv format:

## DefensePro Best Practice Configuration Recommendations report (dpconfig\_report.csv)

DefensePro Best Practice Configuration Recommendations report includes the following configuration checks:

1. DefensePro has no catchall policy
2. Policy has no security profiles applied
3. Policy is configured two-way
4. Policy is in report mode
5. Policy is disabled
6. Packet reporting is disabled
7. Policy has no BDOS profile applied
8. Policy has no Signature profile applied
9. Signature profile applied on the policy does not include all DoS-All rules
10. DNS Signature profile applied on the DNS policy does not include all DoS-All rules and DNS Services Signatures
11. Policy has no Out of state profile applied
12. Policy has no Connection Limit applied
13. Policy has no SYN Flood profile applied
14. Policy has no ERT Active Attacker Feed profile applied
15. DefensePro has no Heartbeat policy for the Silicom Bypass Switch (if exists)
16. Catchall policy has not the lowest priority
17. Policies distribution across instances is not equal for DefensePro version 7.x
18. BDOS profile is in report mode
19. BDOS profile is not applied on any policy(orphaned)

Report name is “dpconfig\_report.csv”

# Setup

## Requirements

The solution requires python 3.6 and higher

## Libraries/packages in use

json

csv

config

socket

logging

smtplib

email

datetime

os

urllib3

sys

## Instructions and recommendations

1. Place the script folder into the appropriate location on the server

2. Install dependencies and necessary libraries/packages

3. Rename file `config.py example` to ‘config.py’ and set the necessary values.

4. Create empty folders “log”, “Raw Data”, “Reports”.

5. Set up the script to run periodically (optional) though cron (linux) or windows scheduler on windows:

Linux cron example

0 4 \* \* \* /path/to/app/bdos\_monitor.sh #example setting the script to run every day at 4 am.

5. Navigate to the folder containing the script and run

.\main.py

* Runs the script, produces all 4 reports and sends them by email

Script can be run with the following arguments (multiple arguments may be combined together)

.\main.py --use-cache-data

* Script parses previously collected data only (stage 2 only, no data collection)

.\main.py --no-alarm

* Script runs without sending email at the end

.\main.py --test-alarm"

* Script runs test email function to test email server connectivity.

# Components for the script operation

## “config.py”

“config.py” includes all the configurable variables in order to run the script. Below is the list of the configurable variables

VISION\_IP = "1.1.1.1" # APSolute Vision IP

VISION\_USER = "user" # APSolute Vision username

VISION\_PASS = "password" # APSolute Vision password

# Script logging set up parameters

LOG\_FILE\_PATH = "./log/" # folder to save the script logging events

LOG\_ROTATION\_SIZE = 20000000 # Maximum rotation log file size in Bytes after which it will be split to another file

LOG\_ROTATION\_HISTORY = 10 # Maximum amount of log files to keep

SYSLOG\_SERVER = "1.1.1.2" # Syslog server destination IP for sending events through syslog

SYSLOG\_PORT = 514 # Syslog server destination UDP port

# Email set up parameters for sending email with reports

SMTP\_SERVER = "smtp.gmail.com" # SMTP server name

SMTP\_SERVER\_PORT = 587 # SMTP server port

SMTP\_SENDER = 'sender@gmail.com' # Email sender address setting

SMTP\_PASSWORD = radware # Email password (optional)

SMTP\_LIST = ['recepient@radware.com'] # Email address/address list recepient/s(comma separated)

SMTP\_SUBJECT\_PREFIX = "ALARM:DP - " # Email Subject

SMTP\_MSG\_BODY = "This email was automated by the DefensePro monitoring script" # Email message body

## “main.py”

* main.py is a main file which actually runs the script
* By default, the script will generate the report and send it by email.
* The script logs errors through syslog for remote monitor and write to a local log.
* At the end, the script sends an email with generated report attached.

## “vision.py”

This file includes all the instructions how to connect to the APSolute Vision and construct proper API calls to fetch the data

## logging\_helper.py

This file includes all the settings and functions for setting up the logging and email functionality

## dpconfig\_parser.py

“dpconfig\_parser.py” parses the collected data and writes deviations from the best practice into “dp\_config.csv” final report. The checks are listed under the “[Script output](#_Script_Output)” section

# Script operation stages

## Stage 1- Data collection

At stage 1, the data is collected for all registered DefensePro/policies and is written to intermediate files.

### Files necessary for the Best practice configuration review

#### full\_bdosprofconf\_dic.json

This file is being generated once the data collection is complete and it includes all the bdos profiles configuration data.

#### full\_net\_dic.json

This file is being generated once the data collection is complete and it includes all the network classes profiles configuration data.

#### full\_sig\_dic.json

This file is being generated once the data collection is complete and it includes all the signature profiles configuration data.

#### full\_pol\_dic.json

“full\_pol\_dic.json” is generated once the data collection is complete, it stores all the information for all the policies for all the registered DefensePro’s in the APSolute Vision and is used for the further data parsing.

## Stage 2- Data parsing

#### dpconfig\_parser.py

“dpconfig\_parser.py” parses the collected data and writes deviations from the best practice into “dp\_config.csv” final report. The checks are listed under the “[Script output](#_Script_Output)” section

## Stage 3 – email alert

By default, once the script finishes its operation, “dp\_config.csv” report will be sent to the email address/es listed in ”config.py”.