**DefensePro – traffic stats collector**

DefensePro Traffic statistics report

Contents

[ChangeLog 3](#_Toc84344994)

[Overview 5](#_Toc84344995)

[Script Output 5](#_Toc84344996)

[DefensePro Traffic statistics report (traffic\_stats.csv) 5](#_Toc84344997)

[Setup 6](#_Toc84344998)

[Requirements 6](#_Toc84344999)

[Libraries/packages in use 6](#_Toc84345000)

[Instructions and recommendations 6](#_Toc84345001)

[Components for the script operation 7](#_Toc84345002)

[“config.py” 7](#_Toc84345003)

[“main.py” 8](#_Toc84345004)

[“vision.py” 8](#_Toc84345005)

[logging\_helper.py 8](#_Toc84345006)

[Script operation stages 9](#_Toc84345007)

[Stage 1- Data collection 9](#_Toc84345008)

[Stage 2- Data parsing 11](#_Toc84345009)

[Stage 3 – email alert 11](#_Toc84345010)

# ChangeLog

v1.3

- Added running dpconfig data parsing with cache data, similarly to bdos

--use-cache-data

v1.4

- Code revise

V1.6

- get\_traffic\_stats

- Created a separate file to collect the traffic and bdos stats (cuurently bdos traffic utilization and normal baseline last 24 hours)

- Added BDOS stats, BPS, PPS

v1.7

- traffic\_stats\_parser

- Added CPS stats

- Bug fixes (CPS, BPS, PPS) when the value is None

V1.8

- traffic\_stats\_parser

- Added CEC stats

- Rearranged BDOS Normal baseline column to come right after the Traffic Utilization

- config.py

- Added “DURATION” as a configurable setting for the report data collection timeframe. (Aug 2, 2021)

V1.9

* Code optimization
* Added BDOS monitoring IPv6 functionality
* “traffic\_stats.csv” added extra 5% to highest average traffic utilization BPS calculation
* “bdos\_parser.py” – improved bdos data collection only in case policy has BDOS profile on it.

V1.10

* Added sending “traffic\_stats.csv” by email
* “traffic\_stats.csv” changed from extra 5% to extra 10% to highest average traffic utilization BPS calculation

V1.11

* Skipped BDOS data collection if the policy name equals “null” (old v6.x code)
* code optimization

V1.13

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

V2.0

* Split code into a dedicated app for traffic statistics collection

V2.1

* Added creation of empty folders if does not exist “log”, “Raw Data”, “Reports”

Functionality to be added:

# Script progress logging

# Add to traffic stats policies that has 0 traffic

# 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 a report which includes DefensePro highest peak average traffic statistics for every policy CPS/PPS/BPS/CEC/BDOS/DNS (.\Reports\traffic\_stats.csv folder).

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 includes one report in csv format:

## DefensePro Traffic statistics report (traffic\_stats.csv)

“traffic\_stats.csv” report includes highest traffic utilization average for the configurable historical timeframe in days (default is 6 days) for every policy including the following stats

CPS = Connections Per Second per policy per DefensePro

PPS = Packets Per Second per policy per DefensePro

BPS = Traffic utilization in Mbps per policy per DefensePro

BDOS protected protocols and Normal baseline BPS per policy per DefensePro

DNS protected record types and Normal baseline QPS per policy per DefensePro

CEC = Concurrent established Connections per DefensePro All policies combined

# Setup

## Requirements

The solution requires python 3.6 and higher

Part of the standard 3.6 library

## Packages required - part of the standard 3.6 library- no need to install

json

csv

socket

logging

smtplib

email

datetime

os

sys

## Packages in use – requires installation

urllib3

Use the following command in order to install urllib3

**pip install -r requirements.txt**

## Instructions and recommendations

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

2. Install dependencies and necessary libraries/packages

3. Rename ‘config.py example to ‘config.py’ configuration 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 the report and sends it 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

DURATION = 6 # sets the time frame in days for the data collection period. For example 1 = 1 day, 7 = 7 days etc.

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

“config.py example” is attached for the reference. In order to run the script, rename “config.py example” to “config.py” in order to run the script.

## “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.
  + This report includes:
    - Traffic utilization for the last DURATION of days(configurable under config.py file, default is 6 days) for the following stats
      * Mbps total traffic utilization per policy
      * Mbps per BDOS Protocol per policy
      * BDOS Normal baselines per policy
      * CPS (Connections Per Second) total traffic utilization per policy
      * PPS (Packets Per Second) total traffic utilization per policy
      * CEC (Concurrent Established Connections) per DefensePro all policies combined.
* The script logs errors through syslog for remote monitor and write to a local log (.\log\monitor.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 all the necessary data

## logging\_helper.py

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

# 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.

Following files are mandatory in order to collect the BDOS/DNS baselines and traffic stats for low/high baselines detection.

#### .\Requests\BDOStrafficRequest.json

Defines the JSON payload for constructing BDOS data (traffic utilization and BDOS baselines) collection API calls.

#### .\Raw Data\BDOS\_traffic\_report.json

This file is being generated after the successful collection of all BDOS baselines and traffic stats for all registered DefensePro and policies. It includes all the traffic raw data and is used for the data parsing at the later stage. The data collection may take long time (~3 hours) and can generate large file of data ~300MB or more, depending on how large is the scale of the environment, how many DefensePro are registered to Vision and historical data collection setting ( DURATION in days under the config.py file).

#### .\Requests\DNStrafficRequest.json

Defines the JSON payload for constructing DNS data (DNS traffic and baselines) collection API calls.

#### .\Raw Data\DNS\_traffic\_report.json

This file is being generated after the successful collection of all DNS baselines and traffic stats for all registered DefensePro and policies. It includes all the traffic raw data and is used for the data parsing at the later stage. The data collection may take long time (~3 hours) and can generate large file of data ~300MB or more, depending on how large is the scale of the environment, how many DefensePro are registered to Vision and historical data collection setting ( DURATION in days under the config.py file).

#### .\Raw 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\_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.

Following files are required in order to construct the API calls to Vision to collect all the CPS/PPS/BPS/CEC traffic statistics

#### TrafficRequest.json

This file is required in order to construct an API call to Vision to collect BPS (Bits Per Second) and PPS(Packets Per Second) data.

#### TrafficRequestCEC.json

This file is required in order to construct an API call to Vision to collect CEC(Concurrent Established Connections) data.

#### TrafficRequestCPS.json

This file is required in order to construct an API call to Vision to collect CPS(Connections Per Second) data.

The files below are being generated once the traffic statistics data collection is complete. They include raw json data.

#### Traffic\_report\_CEC.json

#### Traffic\_report\_CPS.json

#### Traffic\_report\_PPS.json

#### Traffic\_report\_BPS.json

## Stage 2- Data parsing

#### traffic\_stats\_parser.py

traffic\_stats\_parser.py parses the collected traffic raw reports(Traffic\_report\_CEC.json, Traffic\_report\_CPS.json, Traffic\_report\_PPS.json, Traffic\_report\_BPS.json, BDOS\_traffic\_report.json), calculates and presents the highest average traffic utilization statistics for all DefensePro and all policies in csv format ([traffic\_stats.csv](#_traffic_stats.csv)) as an output

## Stage 3 – email alert

Once the script finishes its operation, report/s will be sent to the email address/es defined in”config.py” as variables by default.