**DefensePro – BDOS Baselines monitoring tool**

DefensePro low/high BDOS baselines monitoring report

Contents

[ChangeLog 2](#_Toc82530399)

[Overview 4](#_Toc82530400)

[Script Output 4](#_Toc82530401)

[1. DefensePro Best Practice Configuration Recommendations report (dpconfig\_report.csv) 5](#_Toc82530402)

[2. DefensePro BDOS baselines that are set too low report (low\_bdos\_baselines.csv) 5](#_Toc82530403)

[How does Low BDOS baselines detection works 5](#_Toc82530404)

[3. DefensePro BDOS baselines that are set too high report (high\_bdos\_baselines.csv) 7](#_Toc82530405)

[4. DefensePro Traffic statistics report (traffic\_stats.csv) 7](#_Toc82530406)

[Components for the script operation 7](#_Toc82530407)

[“config.py” 7](#_Toc82530408)

[“main.py” 8](#_Toc82530409)

[“vision.py” 9](#_Toc82530410)

[logging\_helper.py 9](#_Toc82530411)

[Setup 9](#_Toc82530412)

[Requirements 9](#_Toc82530413)

[Libraries/packages in use 9](#_Toc82530414)

[Instructions and recommendations 10](#_Toc82530415)

[Script operation stages 11](#_Toc82530416)

[Stage 1- Data collection 11](#_Toc82530417)

[Files necessary for the BDOS baselines detection 11](#_Toc82530418)

[Files necessary for the Traffic statistics collection 11](#_Toc82530419)

[Files necessary for the Best practice configuration review 12](#_Toc82530420)

[Stage 2- Data parsing 13](#_Toc82530421)

[Stage 3 – email alert 13](#_Toc82530422)

# ChangeLog

v1.3

- Added running the code with argument

--use-cache-data

- bdos\_parser

- Fixed bug with exceeed ratio 0 (condition where virtual baseline = 0)

- Added Lost stats logging into CSV

- Added "No BDOS traffic on all traffic protocols" logging into CSV

- Added High baselines detection

- bdos\_parser

- High baselines - Improved traffic utilization from average traffic utilization to average utilization based on highest 10 traffic values

V1.8

- traffic\_stats\_parser

- Added “DURATION” as a configurable setting for the report data collection timeframe.

V1.9

* Code optimization
* Added BDOS monitoring IPv6 functionality
* “bdos\_parser.py” – improved bdos data collection only in case policy has BDOS profile on it.

V1.10

* Added sending “high\_bdos\_baselines.csv” by email

V1.11

* Skipped BDOS data collection if the policy name equals “null” (old v6.x code)
* “bdos\_parser.py” code optimization
* Added DNS flood baselines low and high baselines detection
* Added BDOS profile checking – alert if in Report mode
* Added BDOS profile checking – alert if BDOS profile is not applied on any policy

V1.13

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

V2.0

* Split code into BDOS monitoring dedicated app

V2.1

* Added empty folders creation
* BDOS and DNS data collection enhancement
* Do not attempt collecting BDOS and DNS stats if policy state is disabled

V2.2

* New feature- detect UDP Normal baselines if less than 100Mbps(configurable)

Functionality to be added:

# Add progress

# High baselines- enhance high baselines detection criteria (currently x4 times)

# 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 2 reports

1. [DefensePro BDOS baselines that are set too low report (low\_bdos\_baselines.csv)](#_DefensePro_BDOS_baselines)
2. [DefensePro BDOS baselines that are set too high report (high\_bdos\_baselines.csv)](#_DefensePro_BDOS_baselines_1)

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 2 reports in csv format:

## 1. DefensePro BDOS baselines that are set too low report (low\_bdos\_baselines.csv)

“low\_bdos\_baselines.csv” report lists all policies with BDOS baselines that are set too low.

### How does Low BDOS baselines detection works

The script collects historical checkpoints (every 20 minutes) of traffic utilization in bps and BDOS Normal baseline. Historical timeframe is configurable in days. If historical timeframe is configured to 1 day, total of 24 hours \* 60 = 1440 (converted to minutes) / 20 (checkpoint is taken every 20 min) = 72 total checkpoints of traffic statistics is collected.

#### Virtual baseline

The script creates a virtual configurable baseline which is derived from the Normal baseline.

Virtual baseline allows additional sensitivity customization. Virtual baseline can be set to be

1. Lower than an actual Normal Baseline (the Greeen Line) = higher sensitivity, allowing reacting on traffic approaching the Normal Baseline ahead of time, for example at 70%,80%, 90% etc.

2. Higher than an actual Normal baseline = lower sensitivity, allowing to react only when it is really critical and close to the Suspect Edge baseline (Yellow line).

Virtual baseline can be set using “DET\_MARGIN\_RATIO” variable in “config.py” file. For example

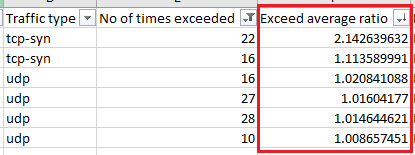
Given margin\_ratio = 0.6, data interval is 20 minutes, and exceed\_threshold = 10

Next step is the script counts occurrences where the actual traffic crossed the virtual baseline. If the number of such traffic-exceeds-normal-edge events count in the defined time period (default 24H ) is greater than the defined threshold (default is 10 occurances), the policy and protocol type will be added to the final report list of policies with low baselines.

#### Exceed average Ratio

The degree of how bad the low baselines are can be measured by the “Exceed average ratio” between traffic and the normal edge when the traffic-exceeds-normal-edge events. This ratio is calculated based on the average of all exceeding events during the defined time period (default 24 hours).

The higher the ratio, the worse the base line. In the example below we can see that the worst “Exceed average ratio” for the “tcp-syn” type of traffic is 2.14 meaning that based on 22 times the tcp-syn traffic exceed the Normal baseline on average 2.14 times or by ~114%



In order to continuously monitor the BDOS baselines, it is recommended to run the main script on a daily basis and collect BDOS Baselines from Radware Vision and traffic statistics for the last 24 hours.

## 2. DefensePro BDOS baselines that are set too high report (high\_bdos\_baselines.csv)

“high\_bdos\_baselines.csv” includes policies and protected protocol types where BDOS baselines are set too high. The evaluation criteria is based on the percentage ratio of the actual traffic to a Normal BDOS baseline (Green line) as well as the Mbits ratio.

# New Feature Requests

* Add dates timeframe as a column into the final report (Hermann, Bell Nov 26, 2021)

# 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

DET\_MARGIN\_RATIO = 1 #sets the virtual baseline ratio. For example 1 = Virtual baseline is the same as the actual Normal BDOS baseline – green line (100% of the actual Normal bassline), 0.8 = 80% of the Normal baseline etc.

DET\_ALARM\_THRESHOLD = 10 #sets threshold for the number of occurances where the actual traffic went above the Virtual baseline. In case there were more occurrences than the defined threshold, these policies and protocols will be listed in the “low\_bdos\_baselines.csv”

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

## “main.py”

* main.py is a main file which actually runs the
* By default, the script will generate the reports and send them by email.
  + Low BDOS baselines report (low\_bdos\_baselines.csv)
    - This reports includes only policies which are considered too low.
  + High BDOS baselines (high\_bdos\_baselines.csv)
    - This report includes only policies which are considered too high
* 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 reports 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

# Setup

## Requirements

The solution requires python 3.6 and higher

## 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 configuration file `config.py example` to ‘config.py’

4. 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.

# 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 BDOS baselines detection

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

#### BDOStrafficRequest.json

Defines the JSON payload for constructing BDOS data collection API calls.

#### 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 big is the environment, how many DefensePro are registered to Vision and historical data collection setting ( DURATION in days in config.py file).

#### DNSStrafficRequest.json

Defines the JSON payload for constructing DNS data collection API calls.

#### 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 big is the environment, how many DefensePro are registered to Vision and historical data collection setting ( DURATION in days in config.py file).

Following files are required in order to construct the API calls to Vision

#### 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.

## Stage 2- Data parsing

#### bdos\_parser.py

This file parses “BDOS\_traffic\_report.json” and “DNS\_traffic\_report.json” raw data that has been generated in the data collection stage, applies conditions and creates final reports as an output, which include all the policies where BDOS baselines were set too low or too high.

high\_bdos\_baselines.csv

low\_bdos\_baselines.csv

## Stage 3 – email alert

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