nagios$_{c}heck_{p}aloaltoDocumentation$

Release 0.3.2

Ralph Offinger

May 16, 2017

Contents

Contents:

CHAPTER 1

nagios_check_paloalto: a Nagios/Icinga Plugin

nagios_check_paloalto is a **Nagios/Icinga plugin** for Palo Alto Next Generation Firewalls. It is written in Python and based on the PA REST API.

Tested on:

- PA-500 v6.0.1 - v6.0.9
- PA-3050 v6.0.9 - 7.1.9

Documentation

http://nagios-check-paloalto.readthedocs.org/en/latest/

Quickstart

Please make sure you have python-dev and libxslt1-dev installed on your machine.

To install nagios_check_paloalto:

```
$ pip install check_paloalto --upgrade
```

or use:

```
$ pip3 install check_paloalto --upgrade
```

The plugin requires a token to get information from the PA-REST-API. Please see the following link for more information: http://nagios-check-paloalto.readthedocs.org/en/latest/configuration.html#token

Usage

Command-line usage:

```
usage: check_paloalto [-h] -H HOST -T TOKEN [-v] [-t TIMEOUT] [--reset]
                      [--version]
                      {diskspace,certificates,load,useragent,environmental,sessinfo,
↪thermal,throughput}
                      ...

positional arguments:
  {diskspace,certificates,load,useragent,environmental,sessinfo,thermal,throughput}
    diskspace           check used diskspace.
    certificates        check the certificate store for expiring certificates:
                        Outputs is a warning, if a certificate is in range.
    load                check the CPU load.
    useragent           check for running useragents.
    environmental       check if an alarm is found.
    sessinfo            check important session parameters.
    thermal             check the temperature.
    throughput          check the throughput.

optional arguments:
  -h, --help            show this help message and exit

Connection:
  -H HOST, --host HOST  PaloAlto Server Hostname
  -T TOKEN, --token TOKEN
                        Generated Token for REST-API access

Debug:
  -v, --verbose         increase output verbosity (use up to 3 times)
  -t TIMEOUT, --timeout TIMEOUT
                        abort check execution after so many seconds (use 0 for
                        no timeout)
  --reset               Deletes the cookie file for the throughput check.

Info:
  --version             show program's version number and exit
```

CHAPTER 2

Installation

Simply install check_paloalto:

```
$ pip install check_paloalto
```

or use:

```
$ pip3 install check_paloalto
```

CHAPTER 3

Configuration

Token

The REST API requires a token to get information. This token must be generated once:

```
1. Create a "monitoring role" in the PA.
2. Disable everything in the WEB UI tab within that role
3. Enable "Operational requests" in the XML API tab and disable everything else
4. Ensure that the tab "Command line" is "None"
5. Create a new Admin user who uses that custom role and for best practices choose at␣
↪least 20 length password without special characters other than '_-'
6. Generating the token is easy. To do that login to your PA with the monitoring user

and open:
https://x.x.x.x/api/?type=keygen&user=YOUR-USERNAME&password=YOUR-PASSWORD
(replace YOUR-USERNAME with the username created in step 5. and YOUR-PASSWORD␣
↪accordingly)
```

Nagios

```
define command {
       command_name    check_paloalto
       command_line    /usr/local/bin/check_paloalto
       }
```

CHAPTER 4

Usage

Command-line usage:

```
usage: check_paloalto [-h] -H HOST -T TOKEN [-v] [-t TIMEOUT] [--reset]
                      [--version]
                      {diskspace,certificates,load,useragent,environmental,sessinfo,
↪thermal,throughput}
                      ...

positional arguments:
  {diskspace,certificates,load,useragent,environmental,sessinfo,thermal,throughput}
    diskspace           check used diskspace.
    certificates        check the certificate store for expiring certificates:
                        Outputs is a warning, if a certificate is in range.
    load                check the CPU load.
    useragent           check for running useragents.
    environmental       check if an alarm is found.
    sessinfo            check important session parameters.
    thermal             check the temperature.
    throughput          check the throughput.

optional arguments:
  -h, --help            show this help message and exit

Connection:
  -H HOST, --host HOST  PaloAlto Server Hostname
  -T TOKEN, --token TOKEN
                        Generated Token for REST-API access

Debug:
  -v, --verbose         increase output verbosity (use up to 3 times)
  -t TIMEOUT, --timeout TIMEOUT
                        abort check execution after so many seconds (use 0 for
                        no timeout)
  --reset               Deletes the cookie file for the throughput check.
```

```
Info:
  --version             show program's version number and exit
```

To check your Palo Alto Firewall, there are several commands available.

diskspace

usage:

```
usage: check_paloalto diskspace [-h] [-w WARN] [-c CRIT]

optional arguments:
  -h, --help            show this help message and exit
  -w WARN, --warn WARN  Warning if diskspace is greater. (default: 85)
  -c CRIT, --crit CRIT  Critical if disksace is greater. (default: 95)
```

example:

```
$ check_paloalto -H HOST -T TOKEN diskspace
$ DISKSPACE OK - sda2: 70%, sda5: 51%, sda6: 58%, sda8: 78% | sda2=70%;85;95 sda5=51%;
↪85;95 sda6=58%;85;95 sda8=78%;85;95
```

certificates

usage:

```
usage: check_paloalto certificates [-h] [-ex EXCLUDE] [-r RANGE]

optional arguments:
  -h, --help            show this help message and exit
  -ex EXCLUDE, --exclude EXCLUDE
                        Exclude certificates from check by name.
  -r RANGE, --range RANGE
                        Warning if days until certificate expiration is in
                        range: Represents a threshold range. The general
                        format is "[@][start:][end] (default: 0:20)
```

example:

```
$ check_paloalto -H HOST -T TOKEN certificates
$ CERTIFICATE WARNING - Certificate1 expires in 8 days
```

load

usage:

```
usage: check_paloalto load [-h] [-w WARN] [-c CRIT]

optional arguments:
  -h, --help            show this help message and exit
```

```
  -w WARN, --warn WARN  Warning if CPU load is greater. (default: 85)
  -c CRIT, --crit CRIT  Critical if CPU load is greater. (default: 95)
```

example:

```
$ check_paloalto -H HOST -T TOKEN load
$ LOAD OK - CPU0: 0.0%, CPU1: 1.0%, CPU2: 4.0%, CPU3: 5.0%, CPU4: 6.0%, CPU5: 5.0% |␣
↪CPU0=0.0%;85;95;0;100 CPU1=1.0%;85;95;0;100 CPU2=4.0%;85;95;0;100 CPU3=5.0%;85;95;0;
↪100 CPU4=6.0%;85;95;0;100 CPU5=5.0%;85;95;0;100
```

environmental

usage:

```
usage: check_paloalto environmental [-h]

optional arguments:
  -h, --help  show this help message and exit
```

example:

```
$ check_paloalto -H HOST -T TOKEN environmental
$ ENVIRONMENTAL OK - No alarms found.
```

sessinfo

usage:

```
usage: check_paloalto sessinfo [-h]

optional arguments:
  -h, --help  show this help message and exit
```

example:

```
$ check_paloalto -H HOST -T TOKEN sessinfo
$ SESSINFO OK - Active sessions: 6582 / Throughput (kbps): 24304 | session=6582;20000;
↪50000;0;262142 throughput_kbps=24304;;;0
```

thermal

usage:

```
usage: check_paloalto thermal [-h] [-w WARN] [-c CRIT]

optional arguments:
  -h, --help            show this help message and exit
  -w WARN, --warn WARN  Warning if temperature is greater. (default: 40)
  -c CRIT, --crit CRIT  Critical if temperature is greater. (default: 45)
```

example:

```
$ check_paloalto -H HOST -T TOKEN thermal
$ THERMAL OK - Temperature @ Ocelot is 29 degrees Celsius, Temperature @ Switch is 33␣
↪degrees Celsius, Temperature @ Cavium is 36 degrees Celsius, Temperature @ Intel␣
↪PHY is 24 degrees Celsius | 'Temperature @ Cavium'=36.5;40;45;5.0;60.0 'Temperature␣
↪@ Intel PHY'=24.2;40;45;5.0;60.0 'Temperature @ Ocelot'=29.9;40;45;5.0;60.0
↪'Temperature @ Switch'=33.8;40;45;5.0;60.0
```

throughput

usage:

```
usage: check_paloalto throughput [-h] -i [INTERFACE]

optional arguments:
  -h, --help            show this help message and exit
  -i [INTERFACE], --interface [INTERFACE]
                        PA interface name, seperate by comma.
```

example:

```
$ check_paloalto -H HOST -T TOKEN throughput -i ethernet1/1
$ THROUGHPUT OK - Input is 5.74 Mb/s - Output is 11.81 Mb/s | 'in_bps_ethernet1/1
↪'=5743432.0;;;0 'out_bps_ethernet1/1'=11807524.0;;;0

$ check_paloalto -H HOST -T TOKEN throughput -i ethernet1/1,ethernet1/2
$ THROUGHPUT OK - Input is 44.12 Mb/s - Output is 24.59 Mb/s | 'in_bps_ethernet1/1
↪'=5895616.0;;;0 'in_bps_ethernet1/2'=38225768.0;;;0 'out_bps_ethernet1/1'=15926620.
↪0;;;0 'out_bps_ethernet1/2'=8661100.0;;;0
```

To get all available names of your interfaces, please have a look at https://www.paloaltonetworks.com/documentation/61/pan-os/pan-os/getting-started/configure-interfaces-and-zones.html

useragents

usage:

```
usage: check_paloalto useragent [-h] [-w WARN] [-c CRIT]

optional arguments:
  -h, --help            show this help message and exit
  -w WARN, --warn WARN  Warning if agent is not responding for a given amount
                        of seconds. (default: 60)
  -c CRIT, --crit CRIT  Critical if agent is not responding for a given amount
                        of seconds. (default: 240)
```

example:

```
$ check_paloalto -H HOST -T TOKEN useragent
$ USERAGENT OK - All agents are connected and responding. | 'Agent: Agent1 -␣
↪HOST1(vsys: vsys1) Host: 192.168.1.1(192.168.1.1):5007'=1;60;240
```

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/ralph-hm/nagios_check_paloalto/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

Write Documentation

nagios_check_paloalto could always use more documentation, whether as part of the official nagios_check_paloalto docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/ralph-hm/nagios_check_paloalto/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *nagios_check_paloalto* for local development.

1. Fork the *nagios_check_paloalto* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/nagios_check_paloalto.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv nagios_check_paloalto
$ cd nagios_check_paloalto/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ make test-all
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Virtual development environment

You can use Vagrant to start a virtual development environment, where all the necessary dependencies are already installed.

Please have a look at the Vagrantfile.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, and 3.4, and for PyPy. Check https://travis-ci.org/ralph-hm/nagios_check_paloalto/pull_requests and make sure that the tests pass for all supported Python versions.

CHAPTER 6

Credits

Development Lead

- Ralph Offinger <ralph.offinger@gmail.com>

Contributors

- Thomas Fischer <mail@se-di.de>

CHAPTER 7

History

CHAPTER 8

0.3.2 (2017-16-05)

- Fixed issue #8: Measuring throughput on multiple identically configured PA's fails

CHAPTER 9

0.3.1 (2017-10-03)

- Improvement: It is no longer necessary to reset the internal cookie when upgrading PA.
- Renamed performance data label for throughput command.
- Removed the the unit of measurement (UOM) for throughput command

CHAPTER 10

0.3 (2017-27-02)

- Support for Python 3.5 added
- Minor code improvements
- Changed the the unit of measurement (UOM) according to official Nagios-Documentation (thanks to Ios77)

CHAPTER 11

0.1.6 (2016-06-05)

- Added script version switch
- Improved error handling
- Updated documentation
- Upgraded dependencies

CHAPTER 12

0.1.5 (2016-29-04)

- Fixed a argparse bug

CHAPTER 13

0.1.4 (2016-29-04)

- Added functionality to monitor state of the user-agents
- Added script timeout switch
- Improved error handling
- Improved functionality of sessinfo command

CHAPTER 14

0.1.3 (2015-14-09)

- Disabled warnings for insecure requests to support older installations: https://urllib3.readthedocs.org/en/latest/security.html

CHAPTER 15

0.1.2 (2015-14-09)

- Fixed a bug for parsing args in python3.
- Enabled warnings for insecure requests: https://urllib3.readthedocs.org/en/latest/security.html
- Changed format for setup.cfg.
- Updated docs.

CHAPTER 16

0.1.1 (2015-10-09)

- Support Python 2.7, 3.3, 3.4.
- Support PyPi.
- Included tests.
- Improved performance.
- Improved output and debugging.

CHAPTER 17

Indices and tables

- genindex
- modindex
- search