AWSIoTDeviceDefenderAgentSDK

Release 1.0

Aug 01, 2018

Contents

1 AWS IoT Device Defender Agent SDK (Python) 1
- 1.1 Prerequisites . . . 1
- 1.2 Notes on the sample agent implementation . . . 1
- 1.3 Quickstart . . . 2
- 1.4 AWS IoT Greengrass Integration . . . 2
- 1.5 Metrics Report Details . . . 5
- 1.6 References . . . 9
- 1.7 License . . . 9
- 1.8 Support . . . 9

2 AWSIoTDeviceDefenderAgentSDK 11
- 2.1 AWSIoTDeviceDefenderAgentSDK . . . 11

3 Indices and tables 13

Python Module Index 15

CHAPTER 1

AWS IoT Device Defender Agent SDK (Python)

Example implementation of an AWS IoT Device Defender metrics collection agent, and other Device Defender Python samples.

The provided sample agent can be used as a basis to implement a custom metrics collection agent.

1.1 Prerequisites

1.1.1 Minimum System Requirements

The Following requirements are shared with the AWS IoT Device SDK for Python

- Python 2.7+ or Python 3.3+ for X.509 certificate-based mutual authentication via port 8883 and MQTT over WebSocket protocol with AWS Signature Version 4 authentication
- Python 2.7.10+ or Python 3.5+ for X.509 certificate-based mutual authentication via port 443
- OpenSSL version 1.0.1+ (TLS version 1.2) compiled with the Python executable for X.509 certificate-based mutual authentication

1.1.2 Connect your Device to AWS IoT

If have never connected your device to AWS IoT before, please follow the Getting Started with AWS IoT Guide. Make sure you note the location of your certificates, you will need to provide the location of these to the Device Defender Sample Agent.

1.2 Notes on the sample agent implementation

client id: The sample agent requires that the client id provided matches a "thing name" in your AWS IoT account. This only for the sake of making the sample easy to get started with. Device Defender only requires that metrics be published for things that are registered in your account, and does not impose any additional requirements on client id

beyond those of the AWS IoT Platform. To customize this behavior, you can modify the way the agent generates the MQTT topic for publishing metrics reports, to use a value other than client id as the thing name portion of the topic.

metric selection: The sample agent attempts to gather all supported Device Defender metrics. Depending on your platform requirements and use case, you may wish to customize your agent to a subset of the metrics.

1.3 Quickstart

1.3.1 Installation

1. Clone the repository

```
git clone https://github.com/aws-samples/aws-iot-device-defender-agent-sdk-python.git
```

1. Install Using pip

Pip is the easiest way to install the sample agent, it will take care of installing dependencies

```
pip install /path/to/sample/package
```

1.3.2 Running the Sample Agent

```
python agent.py --endpoint your.custom.endpoint.amazonaws.com  --rootCA /path/to/
↪rootca  --cert /path/to/device/cert --format json -i 300
```

Command line options

To see a summary of all commandline options:

```
python agent.py --help
```

Test Metrics Collection Locally

```
python collector.py -n 1 -s 1
```

1.4 AWS IoT Greengrass Integration

1.4.1 Overview

AWS IoT Device Defender can be used in conjunction with AWS Greengrass. Integration follows the standard Greengrass lambda deployment model, making it easy to add AWS IoT Device Defender security to your Greengrass Core devices.

1.4.2 Prereqs

1. Greengrass environment Setup
2. Greengrass core configured and running
3. Ensure you can successfully deploy and run a lambda on your core

1.4.3 Using Device Defender with Greengrass Core devices

Create Your Lambda Package

For this portion will be following the general process outlined here

1. Clone the AWS IoT Device Defender Python Samples Repository

```
git clone https://github.com/aws-samples/aws-iot-device-defender-agent-sdk-python.
↪git
```

2. Create, and activate a virtual environment (optional, recommended)

```
pip install virtualenv
virtualenv metrics_lambda_environment
source metrics_lambda_environment/bin/activate
```

3. Install the AWS IoT Device Defender sample agent in the virtual environment Install from PyPi

```
pip install AWSIoTDeviceDefenderAgentSDK
```

Install from downloaded source

```
cd aws-iot-device-defender-agent-sdk-python
#This must be run from the same directory as setup.py
pip install .
```

4. Create an empty directory to assemble your lambda, we will refer to this as your "lambda directory"

```
mkdir metrics_lambda
cd metrics_lambda
```

5. Complete steps 1-4 from this guide
6. Unzip the Greengrass python sdk into your lamda directory

```
unzip ../aws_greengrass_core_sdk/sdk/python_sdk_1_1_0.zip
cp -R ../aws_greengrass_core_sdk/examples/HelloWorld/greengrass_common .
cp -R ../aws_greengrass_core_sdk/examples/HelloWorld/greengrasssdk .
cp -R ../aws_greengrass_core_sdk/examples/HelloWorld/greengrass_ipc_python_sdk .
```

7. Copy the AWSIoTDeviceDefenderAgentSDK module to the root level of your lambda

```
cp -R ../aws-iot-device-defender-agent-sdk-python/AWSIoTDeviceDefenderAgentSDK .
```

8. Copy the Greengrass agent to the root level of your lambda directory

```
cp ../aws-iot-device-defender-agent-sdk-python/samples/greengrass/greengrass_core_
↪metrics_agent/greengrass_defender_agent.py .
```

9. Customize the Greengrass agent to include the name of your Greengrass Core device, and the desired metrics sample rate
    - Replace `GREENGRASS_CORENAME` with the name of your Greengrass Core
    - Set the `SAMPLE_RATE_SECONDS` to your desired metrics reporting interval *Note: 5 minutes (300 seconds) is the shortest reporting interval supported by AWS IoT Device Defender*
10. Copy the dependencies from your virtual environment or your system, into the the root level of your lambda

```
cp -R ../metrics_lambda_environment/lib/python2.7/site-packages/psutil .
cp -R ../metrics_lambda_environment/lib/python2.7/site-packages/cbor .
```

11. Create your lambda zipfile *Note: you should perform this command in the root level of your lambda directory*

```
rm *.zip
zip -r greengrass_defender_metrics_lambda.zip *
```

Configure and deploy your Greengrass Lambda

1. Upload your lambda zip file
2. Select the Python 2.7 runtime, and enter `greengrass_defender_agent.function_handler` in the Handler field
3. Configure your lambda as a long-lived lambda
4. Configure a subscription from your lambda to the AWS IoT Cloud *Note: For AWS IoT Device Defender, a subscription from AWS IoT Cloud to your lambda is not required*
5. Create a local resource to allow your lambda to collect metrics from the Greengrass Core host
    - Follow the instructions here
    - Use the following parameters:
        - **Resource Name:** `Core Proc`
        - **Type:** `Volume`
        - **Source Path:** `/proc`
        - **Destination Path:** `/host_proc`
        - Group owner file access permission: "Automatically add OS group permissions of the Linux group that owns the resource"
        - Associate the resource with your metrics lambda
6. Deploy your lambda to your Greengrass Group

Troubleshooting

Reviewing AWS IoT Device Defender device metrics using AWS IoT Console

1. Temporarily modify your publish topic in your Greengrass lambda to something such as metrics/test
2. Deploy the lambda
3. Add a subscription to the temporary topic in the "Test" section of the iot console, shortly you should the metrics your Greengrass Core is emitting

1.5 Metrics Report Details

1.5.1 Overall Structure

Long Name	Short Name	Required	Type	Constraints	Notes
header	hed	Y	Object		Complete block required for well-formed report
metrics	met	Y	Object		Complete block required for well-formed report

Header Block

Long Name	Short Name	Required	Type	Constraints	Notes
report_id	rid	Y	Integer		Monotonically increasing value, epoch timestamp recommended
version	v	Y	String	Major.Minor	Minor increments with addition of field, major increments if metrics removed

Metrics Block

TCP Connections

Long Name	Short Name	Parent Element	Required	Type	Constraints	Notes
tcp_connections	tc	metrics	N	Object		
established_connections	ec	tcp_connections	N	List		ESTABLISHED TCP State
connections	cs	established_connections	N	List		
remote_addr	rad	connections	Y	Number	ip:port	ip can be ipv6 or ipv4
local_port	lp	connections	N	Number	>0	
local_interface	li	connections	N	String		interface name
total	t	established_connections	N	Number	>= 0	Number established connections

Listening TCP Ports

Long Name	Short Name	Parent Element	Required	Type	Constraints	Notes
listening_tcp_ports	tp	metrics	N	Object		
ports	pts	listening_tcp_ports	N	List	> 0	
port	pt	ports	N	Number	>= 0	ports should be numbers > 0
interface	if	ports	N	String		Interface Name
total	t	listening_tcp_ports	N	Number	>= 0	

Listening UDP Ports

Long Name	Short Name	Parent Element	Required	Type	Constraints	Notes
listening_udp_ports	up	metrics	N	Object		
ports	pts	listening_udp_ports	N	List	> 0	
port	pt	ports	N	Number	> 0	ports should be numbers > 0
interface	if	ports	N	String		Interface Name
total	t	listening_udp_ports	N	Number	>= 0	

Network Stats

Long Name	Short Name	Parent Element	Required	Type	Constraints	Notes
network_stats	ns	metrics	N	Object		
bytes_in	bi	network_stats	N	Number	Delta Metric, >= 0	
bytes_out	bo	network_stats	N	Number	Delta Metric, >= 0	
packets_in	pi	network_stats	N	Number	Delta Metric, >= 0	
packets_out	po	network_stats	N	Number	Delta Metric, >= 0	

1.5.2 Sample Metrics Reports

Long Field Names

```
{
    "header": {
        "report_id": 1529963534,
        "version": "1.0"
    },
```

(continues on next page)

(continued from previous page)

```
    "metrics": {
        "listening_tcp_ports": {
            "ports": [
                {
                    "interface": "eth0",
                    "port": 24800
                },
                {
                    "interface": "eth0",
                    "port": 22
                },
                {
                    "interface": "eth0",
                    "port": 53
                }
            ],
            "total": 3
        },
        "listening_udp_ports": {
            "ports": [
                {
                    "interface": "eth0",
                    "port": 5353
                },
                {
                    "interface": "eth0",
                    "port": 67
                }
            ],
            "total": 2
        },
        "network_stats": {
            "bytes_in": 1157864729406,
            "bytes_out": 1170821865,
            "packets_in": 693092175031,
            "packets_out": 738917180
        },
        "tcp_connections": {
            "established_connections":{
                "connections": [
                    {
                    "local_interface": "eth0",
                    "local_port": 80,
                    "remote_addr": "192.168.0.1:8000"
                    },
                    {
                    "local_interface": "eth0",
                    "local_port": 80,
                    "remote_addr": "192.168.0.1:8000"
                    }
                ],
                "total": 2
            }
        }
    }
}
```

Short Field Names

```
{
    "h": {
        "rid": 1529963534,
        "v": "1.0"
    },
    "met": {
        "tp": {
            "pts": [
                {
                    "if": "eth0",
                    "pt": 24800
                },
                {
                    "if": "eth0",
                    "pt": 22
                },
                {
                    "if": "eth0",
                    "pt": 53
                }
            ],
            "t": 3
        },
        "up": {
            "pts": [
                {
                    "if": "eth0",
                    "pt": 5353
                },
                {
                    "if": "eth0",
                    "pt": 67
                }
            ],
            "t": 2
        },
        "ns": {
            "bi": 1157864729406,
            "bo": 1170821865,
            "pi": 693092175031,
            "po": 738917180
        },
        "tc": {
            "ec":{
                "cs": [
                    {
                    "li": "eth0",
                    "lp": 80,
                    "rad": "192.168.0.1:8000"
                    },
                    {
                    "li": "eth0",
                    "lp": 80,
                    "rad": "192.168.0.1:8000"
                    }
```

(continues on next page)

(continued from previous page)

```
                    ],
                    "t": 2
                }
            }
        }
}
```

1.6 References

- AWS Lambda: Creating a Deployment Package (Python)
- Monitoring with AWS Greengrass Logs
- Troubleshooting AWS Greengrass Applications
- Access Local Resources with Lambda Functions

1.7 License

This library is licensed under the Apache 2.0 License.

1.8 Support

If you have technical questions about the AWS IoT Device SDK, use the AWS IoT Forum. For any other questions about AWS IoT, contact AWS Support.

CHAPTER 2

AWSIoTDeviceDefenderAgentSDK

2.1 AWSIoTDeviceDefenderAgentSDK

2.1.1 AWSIoTDeviceDefenderAgentSDK.agent

2.1.2 AWSIoTDeviceDefenderAgentSDK.collector

2.1.3 AWSIoTDeviceDefenderAgentSDK.metrics

2.1.4 AWSIoTDeviceDefenderAgentSDK.tags

class AWSIoTDeviceDefenderAgentSDK.tags.**Tags**(*short_names=False*)

Bases: object

Abstract field name selection for metrics reports.

HEADER = ('header', 'hed')

METRICS = ('metrics', 'met')

REPORT_ID = ('report_id', 'rid')

VERSION = ('version', 'v')

TCP_CONN = ('tcp_connections', 'tc')

CONNECTIONS = ('established_connections', 'ec')

REMOTE_ADDR = ('remote_addr', 'rad')

REMOTE_PORT = ('remote_port', 'rp')

LOCAL_PORT = ('local_port', 'lp')

LOCAL_INTERFACE = ('local_interface', 'li')

STATUS = ('status', 's')

LISTENING_TCP_PORTS = ('listening_tcp_ports', 'tp')

LISTENING_UDP_PORTS = ('listening_udp_ports', 'up')

PORTS = ('ports', 'pts')

PORT = ('port', 'pt')

NETWORK_STATS = ('network_stats', 'ns')

BYTES_IN = ('bytes_in', 'bi')

BYTES_OUT = ('bytes_out', 'bo')

PACKETS_IN = ('packets_in', 'pi')

PACKETS_OUT = ('packets_out', 'po')

TOTAL = ('total', 't')

get (*tag*)

header

metrics

report_id

version

tcp_conn

connections

remote_addr

remote_port

local_port

local_interface

listening_tcp_ports

listening_udp_ports

ports

interface_stats

interfaces

bytes_in

bytes_out

packets_in

packets_out

total

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

AWSIoTDeviceDefenderAgentSDK, 11
AWSIoTDeviceDefenderAgentSDK.tags, 11

Index

A

AWSIoTDeviceDefenderAgentSDK (module), 11
AWSIoTDeviceDefenderAgentSDK.tags (module), 11

B

BYTES_IN (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12
bytes_in (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12
BYTES_OUT (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12
bytes_out (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12

C

CONNECTIONS (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 11
connections (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12

G

get() (AWSIoTDeviceDefenderAgentSDK.tags.Tags method), 12

H

HEADER (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 11
header (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12

I

interface_stats (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12
interfaces (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12

L

LISTENING_TCP_PORTS (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 11
listening_tcp_ports (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12
LISTENING_UDP_PORTS (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12
listening_udp_ports (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12
LOCAL_INTERFACE (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 11
local_interface (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12
LOCAL_PORT (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 11
local_port (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12

M

METRICS (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 11
metrics (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12

N

NETWORK_STATS (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12

P

PACKETS_IN (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12
packets_in (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12
PACKETS_OUT (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12
packets_out (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12
PORT (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12
PORTS (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12
ports (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12

R

REMOTE_ADDR (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 11
remote_addr (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12
REMOTE_PORT (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 11
remote_port (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12
REPORT_ID (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 11
report_id (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12

S

STATUS (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 11

T

Tags (class in AWSIoTDeviceDefenderAgentSDK.tags), 11
TCP_CONN (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 11
tcp_conn (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12
TOTAL (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12
total (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12

V

VERSION (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 11
version (AWSIoTDeviceDefenderAgentSDK.tags.Tags attribute), 12