Threat Stack Python Client Documentation

Release 1.1.0

Interactive Intelligence, Inc.

Sep 19, 2017

Contents:

The Threat Stack Python Client is a library which allows Python developers to write software that makes use of Threat Stack REST API.

CHAPTER 1

V1 Client Documentation

Resource Types

- agents
- alerts
- logs
- organizations
- policies

For details on each resource please refer to the Threat Stack API documentation.

Create a new client

Import and create a new Threat Stack client:

```
from threatstack import ThreatStack

client = ThreatStack("<API_KEY>")
```

Additional *optional* parameters can be specified when creating a new client object:

```
# default api_version: 1
# default http request timeout = 120

client = ThreatStack(
    api_key=<"API_KEY>",
    org_id="<ORG_ID>",
    api_version=1,
    timeout=120
)
```

Listing Resources

All resource types support a *list* method which returns a generator. The generator can be used to iterate over the results. NOTE: The *list* method uses the default time range for the resouce (see Time Ranges below).

Listing Agents:

```
agents = client.agents.list()

# iterate over results
for agent in agents:
    print agent["id"]
```

Listing Alerts:

```
alerts = client.alerts.list()
```

Listing Logs:

```
logs = client.logs.list()
```

Listing Organizations:

```
orgs = client.organizations.list()
```

Listing Policies:

```
policies = client.policies.list()
```

Time Ranges

Each resource type has a default date/time range that is applied by the Threat Stack API when listing items. Please refer to the Threat Stack API documentation for additional information on these default date/time ranges. To specify a date/time range other than the default, the **start** and **end** parameters can be used.

Using a date range:

```
alerts = client.alerts.list(start="2017-05-01", end="2017-05-31")
```

Using a date and time range:

```
alerts = client.alerts.list(
    start="2017-05-01 08:00:00",
    end="2017-05-01 09:00:00"
)
```

Using *datetime*:

```
from datetime import datetime, timedelta

end = datetime.now()
start = end - timedelta(hours=1)
alerts = client.alerts.list(start=start, end=end)
```

Pagination

The Agents, Alerts, and Logs resource types support pagination. This allows you to control the page and count values when retrieving a list of items. Please refer to the Threat Stack API documentation for additional information on pagination.

```
# default page: 0
# default count: 20

agents = client.agents.list(page=2, count=10)
```

Field Filters

When listing resources you can limit the fields that are returned from the ThreatStack API for each resource. Please refer to the Threat Stack API documentation for additional information on partial responses.

Only retrieve agent hostname and IP address:

```
agents = client.agents.list(fields=["hostname", "ip_address"])
```

Getting a Specific Resource

All resource types other than Logs and Organizations support the *get* method. This method allows you to retrieve details about a specific resource given its unique identifier. The *get* method returns a Python dict.

Getting an Agent:

```
agent = client.agents.get("<AGENT_ID>")
```

Getting an Alert:

```
alert = client.alerts.get("<ALERT_ID>")
```

Getting a Policy:

```
policy = client.policy.get("<POLICY_ID>")
```

If the resource is **not found** by the ThreatStack API, and empty dict is returned from the client.

```
agent = client.agents.get("<AGENT_ID">)
if agent:
    print agent
else:
    print "Agent not found"
```

Additional Methods

Organizations

The Organizations resource has a *users* method which returns a list of users for a given organization.

Get users for an Organization:

```
users = client.organization.users("<org_id>")
```

Additional Examples

Retrieve list of alert ID's from the last hour and then get details for each alert:

```
from datetime import datetime, timedelta
from threatstack import ThreatStack, ThreatStackAPIError

client = ThreatStack("<API_KEY>")

now = datetime.now()
one_hour_ago = now - timedelta(hours=1)

try:
    alerts = client.alerts.list(start=one_hour_ago, end=now, fields=["id"])

    for alert in alerts:
        details = client.alerts.get(alert["id"])

except ThreatStackAPIError:
    print "The ThreatStack API returned an error response."
```

CHAPTER 2

Installation

Install via **pip**:

```
pip install threatstack
```

CHAPTER 3

Usage

Resource Types

- agents
- alerts
- cve vulnerabilities
- rulesets and rules
- servers

For details on each resource please refer to the Threat Stack API documentation.

Create a new client

Import and create a new Threat Stack client:

```
from threatstack import ThreatStack

client = ThreatStack(api_key="<API_KEY>", org_id="<ORG_ID>")
```

Additional *optional* parameters can be specified when creating a new client object:

```
# default api_version: 2
# default http request timeout = 120

client = ThreatStack(
    api_key=<"API_KEY>",
    org_id="<ORG_ID>",
    api_version=[1 or 2],
    timeout=<SECONDS>
)
```

See 'V1 Client Documentation' for using the V1 ThreatStack API Client.

Time Ranges

Time ranges can be specified when listing resources using the *start* and *end* parameters:

```
now = datetime.now()
one_hour_ago = now - timedelta(hours=1)

alerts = client.alerts.list(start=one_hour_ago.isoformat(), end=now.isoformat())

for alert in alerts:
    details = client.alerts.get(alert["id"])
```

Agents

Listing Agents:

```
agents = client.agents.list()

# iterate over results
for agent in agents:
    print agent["id"]

# query parameters
agents = client.agents.list(type=<"monitor" or "investigate">, hostname=<"hostname">,␣
↪instanceId="<instanceId">)
```

Offline Agents:

```
By default only online agents will be listed. To get offline agents set offline to␣
↪True.

offline_agents = client.agents.list(offline=True)
```

Get agent by ID:

```
agent = client.agents.get("<agent_id>")
```

Alerts

Listing Alerts:

```
alerts = client.alerts.list(severity=<1, 2, or 3>)
```

Get Alert by ID:

```
alert = client.alerts.get("<alert_id>")
```

List all dismissed alerts:

```
dismissed = client.alerts.list(dismissed=True)
```

Get severity counts for alerts:

```
sev_counts = client.alerts.severity_counts()
```

Get an event for an alert:

```
event = client.alerts.event(alert_id="<alert_id>", event_id="<event_id>")
```

CVE Vulnerabilities

List all vulnerabilities:

```
vulns = client.vulnerabilities.list()

# query parameters
vulns = client.vulnerabilities.list(package="<package_name>", server="<instanceId/
↪hostname>", agent="<agentId>")
```

Get a vulnerability by CVE number:

```
cve = client.vulnerabilities.get("<cve_number>")
```

List all suppressed vulnerabilities:

```
vulns = client.vulnerabilties.list(suppressed=True)

# same query params can be used as with list
vulns = client.vulnerabilities.list(suppressed=True, package="<package_name>", server=
↪"<instanceId/hostname>", agent="<agentId>")
```

Rulesets & Rules

List all rulesets:

```
rulesets = client.rulesets.list()

# query parameters
rulesets = client.rulesets.list(agentId="<agent_id>")
```

Get a ruleset:

```
ruleset = client.ruleset.get("<ruleset_id>")
```

Get all rules for a ruleset:

```
rules = client.ruleset.rules(ruleset_id="<ruleset_id>")
```

Get a rule for a ruleset:

```
rule = client.ruleset.rules(ruleset_id="<ruleset_id>", rule_id="<rule_id>")
```

Servers

Get all servers:

```
servers = client.servers.list()
```

Get all non-monitored servers:

```
non_monitored = client.servers.list(non_monitored=True)
```