python-nessus-client Documentation

Release 0.1.1

Lukasz Banasiak

Sep 27, 2017

Contents

CHAPTER 1

Overview

Python Client for Nessus 5.0 REST API.

Nessus is a proprietary comprehensive vulnerability scanner which is developed by Tenable Network Security. It is free of charge for personal use in a non-enterprise environment.

CHAPTER 2

Contents:

Installation

You can install python-nessus-client using pip (which is the canonical way to install Python packages).

To install using pip:

```
pip install python-nessus-client
```

Examples

REST resources are translated to methods.

For example:

Resource	Method
`/users/list`	`object.users.list()`
`/server/securesettings/&proxy%5Fport=8888`	`object.server.securesettings(proxy_port='8888')`

and so on...

To get users list `https://nessus.example.com:8834/users/list` we call *`list()`* method on *`Users`* class

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> print nessus.users.list()
[
  {
    "admin": "TRUE",
    "name": "test",
    "lastlogin": 1416492416
```

```
  }
]
```

To get server security settings list `https://nessus.example.com:8834/server/securesettings/list` we call *securesettings()* method on *Server* class

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> print nessus.server.securesettings()
{
  "proxysettings": {
    "proxy_password": null,
    "proxy_port": "8080",
    "custom_host": null,
    "proxy_username": null,
    "user_agent": null,
    "proxy": "10.0.0.1"
  }
}
```

To set server security settings `https://nessus.example.com:8834/server/securesettings/&proxy%5Fport=8888` we use the same *securesettings()* method on *Server* class but we pass as a argument settings to set up.

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> nessus.server.securesettings(proxy_port='8888')
>>> print nessus.server.securesettings()
{
  "proxysettings": {
    "proxy_password": null,
    "proxy_port": "8888",
    "custom_host": null,
    "proxy_username": null,
    "user_agent": null,
    "proxy": "10.0.0.1"
  }
}
```

More examples can be found in the following subsections and in class documentation:

- *API*
- *Server*
- *Users*
- *Plugins*
- *Policy*
- *Scan*
- *Report*

Authenticating a user

Login to Nessus server

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
```

Response is Python structure

We can acts like we work with dict.

Get configuration value

```
>>> print nessus.server.securesettings()['proxysettings']['proxy_port']
8080
```

Get name from second item in report list get list of hosts contained in a specified report

```
>>> second_host = nessus.report.list()[1]['name']
>>> print nessus.report.hosts(second_host)
{
  "scanprogresscurrent": "0",
  "scanprogresstotal": "100",
  (...)
}
```

Make output more readable

Before

```
>>> print nessus.server.securesettings()
{u'proxysettings': {u'proxy_password': None, u'proxy_port': u'8080', (...)
```

After

```
>>> import json
>>> data = nessus.server.securesettings()
>>> json.dumps(data, indent=2)
{
  "proxysettings": {
    "proxy_password": null,
    "proxy_port": "8080",
    "custom_host": null,
    "proxy_username": null,
    "user_agent": null,
    "proxy": "10.0.0.1"
  }
}
```

API

class nessus.**API**(*base_url*, *username=''*, *password=''*, *login=True*, *debug=False*)

Main API class

Parameters

- **base_url** – IP:PORT or FQDN:PORT of Nessus Server

- **username** – user login
- **password** – user password
- **login** (*bool*) – disable autologin to Nessus Server
- **debug** (*bool*) – enable DEBUG mode

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', login=False)
>>> nessus.login('user', 'pass')
```

Is equivalent of:

```
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
```

feed()

Current plugin feed information from the server.

This will return the feed type (HomeFeed vs. ProfessionalFeed), Nessus version and integrated web server version.

Permissions:

•authenticated: Yes

•administrator: No

Returns Feed information.

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> print nessus.feed()
```

get_cert()

Nessus server certificate.

Permissions:

•authenticated: Yes

•administrator: No

Returns Server certificate

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> print nessus.get_cert()
```

login(*login*, *password*)

Authenticates a user.

Permissions:

•authenticated: No

•administrator: No

Parameters

- **login** – user login
- **password** – user password

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', login=False)
>>> nessus.login('user', 'pass')
```

logout()

Log out a user.

It invalidates the token and performs some "house-cleaning" tasks such as deleting the temporary files created for that user.

Permissions:

•authenticated: Yes

•administrator: No

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> nessus.logout()
```

timezones()

Lists time zones that can be specified in a scheduled scan policy.

Permissions:

•authenticated: Yes

•administrator: No

Returns List of time zones

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> print nessus.timezones()
```

uuid()

Nessus server UUID.

Permissions:

•authenticated: Yes

•administrator: No

Returns Server information

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> print nessus.uuid()
```

Server Management

Server

class nessus.**Server**(*uri*, *api*)

load()
Requests the current Nessus server load and platform type.

Permissions:

- authenticated: Yes
- administrator: No

preferences(***kwargs*)
Requests or update the Nessus server advanced settings.

Parameters **kwargs** – settings name and value to change (e.g. checks_read_timeout=5)

Permissions:

- authenticated: Yes
- administrator: Yes

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> print nessus.server.preferences()
{
  "xmlrpc_listen_port": "8834",
  "auto_update_delay": "24",
  "nasl_log_type": "none",
  "log_whole_attack": "no",
  "optimize_test": "yes",
  (...)
}
>>> nessus.server.preferences(xmlrpc_listen_port=8845)
>>> print nessus.server.preferences()
{
  "xmlrpc_listen_port": "8845",
  "auto_update_delay": "24",
  "nasl_log_type": "none",
  "log_whole_attack": "no",
  "optimize_test": "yes",
  (...)
}
>>> print nessus.server.preferences()['xmlrpc_listen_port']
8845
```

register(*code*)
Registers the Nessus server with Tenable Network Security using the plugin feed registration code.

Parameters **code** – a Nessus plugin feed registration code

Permissions:

•authenticated: No

•administrator: No

restart()
Directs the Nessus server to restart.

This function is only valid during the initial installation and registration process.

Permissions:

•authenticated: No

•administrator: No

securesettings(***kwargs*)
Requests or update the Nessus server settings

Proxy information, User-Agent, and custom update host.

Parameters **kwargs** – settings name and value to change (e.g. `proxy='example.com'`)

Permissions:

•authenticated: Yes

•administrator: Yes

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> print nessus.server.securesettings()
{u'proxysettings': {u'proxy_password': None, u'proxy_port': u'8080', (...)
>>> nessus.server.securesettings(proxy_port='8081')
>>> print nessus.server.securesettings()
{u'proxysettings': {u'proxy_password': None, u'proxy_port': u'8081', (...)
>>> print nessus.server.securesettings()['proxysettings']['proxy_port']
8081
```

update()
Directs the Nessus server to force a plugin update.

Note that if the server is not yet registered, then authentication is not required. Once the server is registered with a Nessus Feed ID, then the request must be made as an authenticated administrator.

Permissions:

•authenticated: Yes

•administrator: Yes

Preferences

class nessus.**Preferences**(*uri*, *api*)

list()
List of settings from the `nessusd.conf` file.

Permissions:

- authenticated: Yes
- administrator: No

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> print nessus.preferences.list()
{
  "listen_port": "1241",
  "max_hosts": "80",
  "auto_update": "yes",
  "throttle_scan": "yes",
  (...)
}
```

User Management

class `nessus.`**Users**(*uri*, *api*)

add(*login*, *password*, *admin=False*)
Creates a new user in the Nessus user's database.

This effectively creates the user and its home directory on disk. The login must match the regex `^[a-zA-Z0-9.@-]+$`. Only an administrator can create another user.

Parameters

- **login** – name of the user to create
- **password** – password for this user
- **admin** – set to 1 if the new user will be declared as an administrator

Permissions:

- authenticated: Yes
- administrator: Yes

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> print nessus.users.list()
[
  {
    "admin": "TRUE",
    "name": "test",
    "lastlogin": 1416492416
  }
]
>>> nessus.users.add('test2', 'pass2')
>>> print nessus.users.list()
```

```
[
  {
    "admin": "TRUE",
    "name": "test",
    "lastlogin": 1416492416
  },
  {
    "admin": "FALSE",
    "name": "test2"
  }
]
```

Todo

add login regexp verification `^[a-zA-Z0-9.@-]+$`

chpasswd(*password*)

Lets a user or administrators change their password.

Parameters **password** – the user's password to be changed

Permissions:

- authenticated: Yes
- administrator: No

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> nessus.users.chpasswd('turbotajnehaslo')
```

delete(*login*)

Deletes an existing user.

Under the hood, this will delete the user home directory (i.e., `/opt/nessus/var/nessus/users/<userName>/`), including this user's policies and reports.

Parameters **login** – name of the user to delete

Permissions:

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> nessus.users.delete('test2')
```

- authenticated: Yes
- administrator: Yes

edit(*login*, *password=None*, *admin=None*)

Edits the details of an existing user.

The user's password and admin status can be modified, however the username cannot be.

Parameters

- **login** – name of the user to edit
- **password** – password of the user
- **admin** – True for yes, False for no

Permissions:

•authenticated: Yes

•administrator: Yes

Example:

Set new password for user test2:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> nessus.users.edit('test2', password='newpass')
```

Make user test2 admin:

```
>>> nessus.users.edit('test2', admin=True)
>>> print nessus.users.list()
[
  (...),
  {
    "admin": "TRUE",
    "name": "test2"
  }
]
```

list()

Lists the users on the Nessus scanner.

The result contains their administrator status and the time they last logged in.

Permissions:

•authenticated: Yes

•administrator: Yes

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> print nessus.users.list()
[
  {
    "admin": "TRUE",
    "name": "test",
    "lastlogin": 1416492416
  }
]
```

Plugin Management

class nessus.**Plugins**(*uri*, *api*)

attributes()

Raises **NotImplementedError** –

Todo

/plugins/attributes/list

---

attributes_list_family_search()

Raises **NotImplementedError** –

Todo

/plugins/attributes/familySearch

---

attributes_list_plugin_search()

Raises **NotImplementedError** –

Todo

/plugins/attributes/pluginSearch

---

description(*fname*)

Description of a given plugin including its cross references and more.

The file name of the plugin (e.g., ping_host.nasl) must be passed as an argument.

Parameters **fname** – the name of the plugin to describe (filename)

Permissions:

- authenticated: Yes
- administrator: No

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> print nessus.plugins.description('aix_U811383.nasl')
{
  "pluginattributes": {
    "description": "The remote host is missing AIX PTF U811383 which is␣
↪related to the security of the package rsct.basic.hacmp.2.3.11.0 You should␣
↪install this PTF for your system to be up-to-date.",
    "plugin_version": "$Revision: 1.4 $",
    "plugin_modification_date": "2011/03/14",
    "solution": "Run ' suma -x -a RqType=Security ' on the remote system",
    "risk_factor": "High",
    "synopsis": "The remote host is missing a vendor supplied security patch",
    "plugin_publication_date": "2008/02/12",
    "plugin_type": "local"
  },
  "pluginid": "30766",
  "pluginname": "AIX 520009 : U811383",
  "pluginfamily": "AIX Local Security Checks"
}
```

descriptions()
List of all plugin descriptions from the Nessus server.

Permissions:

- authenticated: Yes
- administrator: Yes

> **Warning:** This request returns a very large response (e.g., over 10 MB).

list(*family=None*)
List of plugin families loaded by the remote server.

List as well as the number of plugins of each family and list of plugins contained in the family.

Parameters **family** – the plugin family to list

Permissions:

- authenticated: Yes
- administrator: No

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> print nessus.plugins.list()
{
  "Mandriva Local Security Checks": "2871",
  "Windows : Microsoft Bulletins": "948",
  "Netware": "14",
  "Misc.": "911",
  "CGI abuses": "3127",
  "Policy Compliance": "37",
  (...)
}
>>> print nessus.plugins.list(family='AIX Local Security Checks')
[
  {
    "pluginid": "55364",
    "pluginfilename": "aix_U840865.nasl",
    "pluginname": "AIX 530011 : U840865",
    "pluginfamily": "AIX Local Security Checks"
  },
  {
    "pluginid": "54191",
    "pluginfilename": "aix_U837183.nasl",
    "pluginname": "AIX 710000 : U837183",
    "pluginfamily": "AIX Local Security Checks"
  },
  (...)
]
```

md5()
List of plugin file names and corresponding MD5 hashes.

Permissions:

•authenticated: Yes

•administrator: No

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> print nessus.plugins.md5()
[
  "aix_U807831.nasl": "cfb861054ad33224cb9f76cd465cea04",
  "aix_U829081.nasl": "79159fa868a6bf266a004a2abcda08e5",
  "fedora_2004-313.nasl": "cc8281f624420f0d03a530cb015eab89",
  (...)
]
```

preferences()

List of plugin-defined preferences.

Permissions:

•authenticated: Yes

•administrator: No

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> print nessus.plugins.preferences()
[
  {
    "preferencetype": "entry",
    "fullname": "ADSI Settings[entry]:Domain Controller :",
    "preferencename": "Domain Controller :",
    "pluginname": "ADSI Settings",
    "preferencevalues": null
  },
  (...)
]
```

Policy Management

class nessus.**Policy**(*uri*, *api*)

add()

Raises **NotImplementedError** –

Todo

/policy/add

copy(*policy_id*)

Copy an existing policy to a new policy.

Parameters **policy_id** – numeric ID of the policy

Permissions:

- authenticated: Yes
- administrator: No

delete(*policy_id*)

Delete an existing policy.

Parameters **policy_id** – numeric ID of the policy

Permissions:

- authenticated: Yes
- administrator: No

download(*policy_id*)

Download the policy from the Nessus scanner to your local system.

Parameters **policy_id** – numeric ID of the policy

Permissions:

- authenticated: Yes
- administrator: Yes

edit()

Raises **NotImplementedError** –

Todo

/policy/edit

---

list()

List of available policies, policy settings and the default values that would be used when creating a new Nessus scan.

The list of default values are the values that will be used during a scan if they are not supplied by the user in the policy (taken from nessusd.rules).

For example, you could save a policy with only one item in it (e.g., `max_checks = 42`) and the rest of the settings used for the scan would be what is returned in *list()*. Custom policies that are returned only include enabled plugins (i.e., disabled plugins will not be returned).

Permissions:

- authenticated: Yes
- administrator: No

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> print nessus.policy.list()
[
  {
    "policyname": "Internal Network Scan",
    "policycontents": {
      "individualpluginselection": {
        "pluginitem": [
```

```
          {
            "status": "enabled",
            "pluginid": "34220",
            "pluginname": "Netstat Portscanner (WMI)",
            "family": "Port scanners"
          },
          (...)
```

upload()

Raises **NotImplementedError** –

Todo

/file/upload, /file/policy/import

Scan Management

class nessus.**Scan**(*uri*, *api*)

list()

List all current scan jobs.

Permissions:

- authenticated: Yes
- administrator: No

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> print nessus.scan.list()
{
  "templates": {},
  "policies": {
    "policies": {
      "policy": [
        {
          "user_permissions": 128,
          "policyName": "Internal Network Scan",
          "policyOwner": "test",
          "policyID": -1,
          "visibility": "shared"
        },
        (...)
      ]
    }
  },
  "scans": {
    "scanList": {
      "scan": []
    }
```

```
  }
}
```

new(*target*, *scan_name*, *policy_id*)
Create a new scan job.

The target parameter is a list, tuple or comma separated string, under any form of target specification (e.g., hostname, IP, range, etc.).

Parameters

- **target** – list, tuple or comma separated string
- **scan_name** – a name for the scan job
- **policy_id** – numeric ID of the policy to use for the scan

Permissions:

•authenticated: Yes

•administrator: No

Note: Once a scan is created, it is assigned a Universally Unique ID (UUID) that will be used on all further requests related to that scan.

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> target = ['localhost', 'example.com']
>>> nessus.scan.new(target, 'test', '-37')
```

pause(*scan_uuid*)
Pause an existing scan job, allowing it to be resumed at a later time.

Parameters **scan_uuid** – UUID of scan job to pause

Permissions:

•authenticated: Yes

•administrator: No

resume(*scan_uuid*)
Resume a previously paused scan job.

Parameters **scan_uuid** – UUID of scan job to resume

Permissions:

•authenticated: Yes

•administrator: No

stop(*scan_uuid*)
Stop an existing scan job.

Parameters **scan_uuid** – UUID of scan job to stop

Permissions:

•authenticated: Yes

•administrator: No

template_delete()

Raises **NotImplementedError** –

Todo

/scan/template/delete

template_edit()

Raises **NotImplementedError** –

Todo

/scan/template/edit

template_launch()

Raises **NotImplementedError** –

Todo

/scan/template/launch

template_new()

Raises **NotImplementedError** –

Todo

/scan/template/new

Reporting

class nessus.**Report**(*uri*, *api*)

attributes(*report*)
List of filter attributes associated with a given report.

Parameters **report** – UUID of the report

Permissions:

•authenticated: Yes

•administrator: No

can_delete_item(*report*)
Determine if a specified report allows items to be deleted.

Parameters **report** – UUID of the report

Permissions:

•authenticated: Yes

•administrator: No

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> name = nessus.report.list()[0]['name']
>>> nessus.report.can_delete_item(name)
True
>>> if nessus.report.can_delete_item(name):
>>>     print 'Report {} allows items to be deleted'.format(name)
Report 95c309f8-2578-fd3e-9e4d-a8aa6d6511e8b617b5a088c93309 allows items to␣
↪be deleted
```

delete (*report*)

Delete a specified report.

Parameters **report** – UUID of the report to be deleted

Permissions:

•authenticated: Yes

•administrator: No

details (*report*, *hostname*, *port*, *protocol*)

Details of a scan for a given host.

Parameters

- **report** – UUID of the report
- **hostname** – name of host to display scan details for
- **port** – port to display scan results for
- **protocol** – protocol of open port on host to display scan details for

Permissions:

•authenticated: Yes

•administrator: No

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> name = nessus.report.list()[0]['name']
>>> print nessus.report.details(name, '127.0.0.1', '0', 'tcp')
[
  {
    "severity": "1",
    "pluginid": "19506",
    "pluginname": "Nessus Scan Information",
    "item_id": "117",
    "data": {
      "description": "This script displays, for each tested host, information␣
↪about the scan itself :(...)",
      "plugin_modification_date": "2014/06/20",
      "plugin_name": "Nessus Scan Information",
      "plugin_publication_date": "2005/08/26",
```

```
      "script_version": "$Revision: 1.69 $",
      "solution": "n/a",
      "risk_factor": "None",
      "synopsis": "Information about the Nessus scan.",
      "fname": "scan_info.nasl",
      "plugin_type": "summary",
      "@xmlns:cm": "http://www.nessus.org/cm"
    },
    "port": "general/tcp"
  },(...)
```

Todo

check if all args are required

download()

Raises **NotImplementedError** –

Todo

/file/report/download, /chapter, /chapter/list, /file/xslt, /file/xslt/list

errors(*report*)

List of any errors associated with a given report.

Parameters **report** – UUID of the report

Permissions:

- authenticated: Yes
- administrator: No

has_audit_trail(*report*)

Determine if a specified report has an Audit Trail associated with it.

Parameters **report** – UUID of the report

Permissions:

- authenticated: Yes
- administrator: No

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> name = nessus.report.list()[0]['name']
>>> nessus.report.has_audit_trail(name)
True
>>> if nessus.report.has_audit_trail(name):
>>>     print 'Report {} has audit trail'.format(name)
Report 95c309f8-2578-fd3e-9e4d-a8aa6d6511e8b617b5a088c93309 has audit trail
```

has_kb(*report*)

Determine if a specified report has a KB associated with it.

Parameters **report** – UUID of the report

Permissions:

- authenticated: Yes
- administrator: No

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> name = nessus.report.list()[0]['name']
>>> nessus.report.has_kb(name)
True
>>> if nessus.report.kb(name):
>>>     print 'Report {} has a KB associated with it'.format(name)
Report 95c309f8-2578-fd3e-9e4d-a8aa6d6511e8b617b5a088c93309 has a KB␣
↪associated with it
```

hosts (*report*)

List of hosts contained in a specified report.

Parameters **report** – UUID of the report

Permissions:

- authenticated: Yes
- administrator: No

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> name = nessus.report.list()
[
  {
    "status": "imported",
    "timestamp": 1416478505,
    "name": "95c309f8-2578-fd3e-9e4d-a8aa6d6511e8b617b5a088c93309",
    "readableName": "Test Scan"
  },
  (...)
]
>>> print nessus.report.hosts('95c309f8-2578-fd3e-9e4d-
↪a8aa6d6511e8b617b5a088c93309')
{
  "scanprogresscurrent": "0",
  "scanprogresstotal": "100",
  "totalchecksconsidered": "100",
  "hostname": "127.0.0.1",
  "numchecksconsidered": "100",
  "severitycount": {
    "item": [
      {
        "severitylevel": "0",
        "count": "0"
      },
      {
        "severitylevel": "1",
        "count": "10"
      },
```

```
      {
        "severitylevel": "2",
        "count": "0"
      },
      {
        "severitylevel": "3",
        "count": "1"
      }
    ]
  },
  "severity": "11"
}
```

Get second host name from list and pass as arg to *hosts()*:

```
>>> second_host = nessus.report.list()[1]['name']
>>> print nessus.report.hosts(second_host)
{
  "scanprogresscurrent": "0",
  "scanprogresstotal": "100",
  (...)
}
```

list()

List of available scan reports.

Permissions:

- authenticated: Yes
- administrator: No

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> print nessus.report.list()
[
  {
    "status": "imported",
    "timestamp": 1416478505,
    "name": "95c309f8-2578-fd3e-9e4d-a8aa6d6511e8b617b5a088c93309",
    "readableName": "Test Scan"
  },
  (...)
]
```

ports(*report*, *hostname*)

List of ports, and the number of findings on each port for each severity.

Severities: Info, Low, Medium, High, Critical

Parameters

- **report** – UUID of the report
- **hostname** – name of host to display open ports for

Permissions:

- authenticated: Yes

•administrator: No

Example:

```
>>> from nessus import API
>>> nessus = API('https://127.0.0.1:8834', username='user', password='pass')
>>> name = nessus.report.list()[0]['name']
>>> print nessus.report.ports(name, '127.0.0.1')
[
  {
    "svcname": "general",
    "portnum": "0",
    "protocol": "tcp",
    "severity": "3",
    "severitycount": {
      "item": [
        {
          "severitylevel": "0",
          "count": "0"
        },
        {
          "severitylevel": "1",
          "count": "2"
        },
        {
          "severitylevel": "2",
          "count": "0"
        },
        {
          "severitylevel": "3",
          "count": "1"
        }
      ]
    }
  }, (...)
```

tags (*report*, *hostname*)

Tags of a scan for a given host.

Some plugins can create "tags" for a remote host that can be extracted later. For example, the OS fingerreturn plugin creates the tag "operating-system" with the actual OS as a value. This makes it easier to extract data automatically.

Parameters

- **report** – UUID of the report
- **hostname** – name of host to display scan details for

Permissions:

•authenticated: Yes

•administrator: No

Note: "Tags" cover plugin-supplied information, such as the OS name, type of credentials used, etc.

---

trail_details (*report*, *hostname*, *plugin_id*)

Audit trail details for a specified report.

Parameters

- **report** – UUID of the report
- **hostname** – host name or IP (optional)
- **plugin_id** – numeric ID of a Nessus plugin

Permissions:

•authenticated: Yes

•administrator: No

Todo

check if all args are required

---

upload(*path*)

Raises **NotImplementedError** –

Todo

`/file/report/import`

CHAPTER 3

Indices and tables

- genindex
- search

Index

T

U