pySSLScan Documentation

Release 0.1

DinoTools

August 02, 2014

Contents

CHAPTER 1

Installation

You can install `pySSLScan` with `pip`:

```
$ pip install sslscan
```

See *Introduction* for more information.

Contents:

1.1 Introduction

1.1.1 Installation

As a Python egg

You can install the most recent `pySSLScan` version using `pip`

```
$ pip install sslscan
```

From a tarball release

Download the most recent tarball from github, unpack it and run the following command on the command-line.

```
$ python setup.py install
```

Install the development version

Install git and run the following commands on the command-line.

```
$ git clone https://github.com/DinoTools/pysslscan.git
$ cd pysslscan
$ python setup.py install
```

1.2 How to use

The pySSLScan framework provides an API to write tests for SSL enabled services. But it also includes a command-line interface to get you started in a few steps.

1.2.1 Command-line

Use the `--help` parameter to display the main help. This will give a short overview about all global options available and list all subcommands.

```
$ pysslscan --help
```

Subcommands are very helpful and extend the command-line interface. To get help for a subcommand just specify the command and append the `--help` option. The result of the following example command will be the help for the `scan` command.

```
$ pysslscan scan --help
```

Performe a basic scan

First of all get a list of all available scan modules.

```
$ pysslscan scan.list
client.ciphers - List all client ciphers.
server.preferred_ciphers - Detect preferred server ciphers.
server.certificate - Extract certificate information.
...
```

After that determine what reporting modules are available.

```
$ pysslscan report.list
term - Print results to the terminal.
...
```

Choose some of the modules and perform a target scan. In the example below two scan modules are used. The first one is `server.ciphers` to detect all supported ciphers available on the server and the second one is `vuln.heartbleed` to run test to detect if the server is vulnerable by the heartbleed bug. To display the scan results on the command-line the reporting module `term` is used. The `--tls10` option enables all TLSv1.0 ciphers.

```
$ pysslscan scan --scan=server.ciphers --scan=vuln.heartbleed --report=term --tls10 127.0.0.1
```

Highlight the result

pySSLScan provides also some rating modules to highlight important facts in the result.

First of all have a look at the list of available rating modules.

```
$ pysslscan rating.list
ssllabs.2009c - Rating used by SSL Labs 2009c
ssllabs.2009d - Rating used by SSL Labs 2009d
...
```

Perform the scan from an earlier example but specify a rating module.

```
$ pysslscan scan --scan=server.ciphers --scan=vuln.heartbleed --report=term:rating=ssllabs.2009e --tl
```

Use a protocol handler

pySSLScan has support for different protocols which are handled by a special handler module. By default pySSLScan will perform a basic TCP connect to scan a target but it supports also protocols like HTTP or SMTP.

The example below will print a list of all available handler modules.

```
$ sslscan.py handler.list
tcp - Handle raw TCP-connections.
smtp - Handle SMTP-connections.
http - Handle HTTP-connections.
...
```

To use a handler module it has to be specified as shown in the next example.

```
$ pysslscan scan --scan=server.ciphers --report=term:rating=rbsec --tls10 'smtp://127.0.0.1:25?startt
```

1.2.2 Python API

ToDo

1.3 API Reference

1.3.1 Scanner

class sslscan.**Scanner**(*module_manager=None*)

The main scanner object.

append(*module*)

Append a scan or report module.

Parameters **module** – Instance of a scan or report module

append_load(*name*, *config*, *base_class=None*)

Append a module but load it first by using the module manager.

Parameters

- **name** (*String*) – Name of the module to load
- **config** (*Mixed*) – Config of the module
- **base_class** (*class*) – Module lookup filter

Returns False if module not found

get_enabled_methods()

Uses the scanner config to create and return a list of all enabled SSL methods.

Returns List of methods

Return type List

get_knowledge_base()

Return the knowledge base used by this scanner.

get_module_manager()

Return the active module manager for this scanner.

load_handler_from_uri(*host_uri*)

Load a handler from a given uri.

Parameters **host_uri** (*String*) – The URI

Returns The handler

load_rating(*name*)
Use the active module manager to load a rating module

Parameters **name** (*String*) – Name of the rating module

run()
Execute all scan and report modules attached to the scanner.

run_reports()
Execute all report modules attached to the scanner.

run_scans()
Execute all scan modules attached to the scanner.

set_handler(*handler*)
Set the active protocol handler.

Parameters **handler** – Instance of the handler

1.3.2 Config

A collection of classes to handle the configuration of a scanner or a module.

class sslscan.config.**BaseConfig**(*options=None*, *parent=None*)
The base config. All other configuration classes use it as base class.

add_option(*name*, ***kwargs*)
Add an option.

Parameters

- **name** (*String*) – Name of the config option
- **kwargs** – Additional params are used for a new sslscan.config.Option instance

add_option_group(*group*)
Add grouped options.

Parameters **group** (sslscan.config.OptionGroup) – Instance of sslscan.config.OptionGroup

get_option(*name*)
Return an option.

Parameters **name** (*String*) – The name of the option

Returns The option or None if not found

get_option_map()
Return the option map

get_option_names()
Return list of option names

get_parent()
Return the parent config object or None if no parent is set.

Returns Object or None

get_value(*name*, *default=None*)
Get the value of an option.

Parameters

- **name** (*String*) – Name of the option

- **default** (*Mixed*) – Default value

Returns If found the value of the option or the default value

set_parent(*parent*)
Set the current parent config object.

Parameters **parent** (*Object|None*) – Set or reset parent config object

set_value(*name*, *value*)
Set the value of an option.

Parameters

- **name** (*String*) – Name of the option
- **value** (*Mixed*) – The value of the option to set

Returns False or True

Return type Boolean

set_values(*data*)
Set the value of multiple options at once.

Parameters **date** – The values to set

Todo Improve docs

class sslscan.config.**ModuleConfig**(*module=None*, ***kwargs*)
Holds the config of a module

Parameters **module** – The module this config is for

get_module()

class sslscan.config.**Option**(*name*, *action='store'*, *default=None*, *help=''*, *metavar=''*, *type='string'*, *values=None*, *negation=None*, *parent=None*)

convert_value_type(*value*)
Tries to convert the value into the right type

Parameters **value** (*Mixed*) – Value to convert

Returns The value

Return type Mixed

get_parent()
Return the parent config object or None if no parent is set.

Returns Object or None

get_value(*default=None*)
Get the value.

Parameters **default** (*Mixed*) – Default value if value of option not set

Returns The value or the default value

Return type Mixed

set_value(*value*)
Set the value and returns True if it was successful or False if not.

Parameters **value** (*Mixed*) – The value

Raises **sslscan.exception.OptionValueError** if types do not match

class sslscan.config.**OptionGroup**(*label*, *help=None*)
Used to group multiple options

class sslscan.config.**ScanConfig**(***kwargs*)
Holds the config of a scanner instance

1.3.3 Knowledge base

The knowledge base is used to store and access all collected information.

Example 1:

```
>>> kb = KnowledgeBase()
>>> kb.set("test.foo", 1234)
>>> kb.get("test.foo")
```

Example 2:

```
>>> kb = KnowledgeBase()
>>> cipher = Cipher()
>>> kb.append("client.ciphers", cipher)
>>> kb.get("client.ciphers")
```

Example 3:

```
>>> group = ResultGroup(label="My Results")
>>> value = ResultValue(label="Yes/No", True)
>>> group.append(value)
```

class sslscan.kb.**BaseResult**(*label=None*)
Base class for custom results.

class sslscan.kb.**Cipher**(*method=None*, *name=None*, *bits=None*, *status=None*)
This class is used to store all information for a cipher.

method_name

status_name

class sslscan.kb.**KnowledgeBase**
The knowledge base is used to store and access all collected information.

append(*kb_id*, *value*)
Append a new value to the knowledge base.

Parameters

- **kb_id** (*String*) – The ID of the value
- **value** (*Mixed*) – The value

get(*kb_id*)
Fetch a value by its ID

Parameters **kb_id** (*String*) – The ID

Todo Add default value

get_group_ids(*kb_id*)
Collect and return all values that are result groups.

The given kb_id is used as filter.

Parameters **kb_id** (*String*) – The ID

get_list(*kb_id*)
Fetch all values and sub-values by a given ID

Parameters **kb_id** (*String*) – The ID

Returns List of values

Return type List

set(*kb_id*, *value*)

class sslscan.kb.**ResultGroup**(***kwargs*)
Group results

append(*item*)

get_items()

class sslscan.kb.**ResultValue**(*name=None*, *value=None*, ***kwargs*)
A single result value

1.3.4 Modules

class sslscan.module.**BaseModule**(*scanner=None*, *config=None*)
Base class used by all modules.

It provides the basic functionality.

get_scanner()
Get the current scanner instance.

set_scanner(*scanner*)
Set the scanner instance the module was appended to.

1.4 Changelog

1.4.1 0.3 - master

Note: This version is not yet released and is under active development.

1.4.2 0.2 - 2014-07-28

- Add: API documentation and docstrings
- Add: Support for Python 2.x
- Add: Logging
- Change: Improve command-line UI

1.4.3 0.1 - 2014-05-11

Proof of concept

- Initial release.

CHAPTER 2

Indices and tables

- *genindex*
- *modindex*
- *search*

S