python-portscan Documentation

Release 3.0

Daniel Thurau

Sep 14, 2017

Contents

CHAPTER 1

About Kali_Port_Scanning

Kali_Port_Scanning is a complex wrapper around the nmap command line utility.

Current modules

- BusinessUnit: Creates and structures data per business unit.
- ScanObject: Holds the scanning information for each IP segment.
- Emailing: Configures push notifications via emails.
- HTMLGenerator: Programatically generates HTML output.
- Log: Allows logging of key features.
- Upload: Allows uploading of reports to DropBox.

Kali_Port_Scanning's modules

The full source code is availible on GitHub.

The different modules are documented below:

BusinessUnit Module

Purpose of BusinessUnit

The BusinessUnit object is the backbone of portscan, it structures the data needed for a business unit to exist, handles reading of input, creates ScanObjects, handles concurrency of scans, and parses output.

Using BusinessUnit

Most modules while callable by the user are integrated into BusinessUnit. An example workflow is given below.

```
from portscan import BusinessUnit

# create a BusinessUnit object with required arugments
BU = BusinessUnit.BusinessUnit('test_Business', '.')

# populate data structures by reading in config files
BU.ReadPorts()
BU.ReadBase()

# Trigger the nmap scans
BU.Scan()

# Collect all nmap data and write to file
BU.Collect()
```

BusinessUnit methods

ScanObject Module

Purpose of ScanObject

The ScanObject module is an object that is owned BusinessUnit. This recieves data from the BusinessUnit reading the config files, and configures the nmap scan to be run. It holds data such as: flags, IP set, ports, and output file. It will configure the final command and serve it to the BusinessUnit object when the BusinessUnit.scan() method is called.

Using ScanObject

ScanObject is implicitly called, but the user still has access to it. The BusinessUnit keeps a list of these objects under BusinessUnit.scan_objs. An basic example of how a ScanObject operates and serves its data is given below.

```
from portscan import ScanObject

...

BU_SO = ScanObject.ScanObject()

BU_SO.CreateCommand("127.0.0.1:22", "-127.0.0.2", "23", "nmap-test")

print(BU_SO.command)

$ open {nmap-dir}/out.html
```

ScanObject methods

class portscan.scanobject.**ScanObject**

__init__()
Contructor of object containing a single nmap scan entity.

CreateCommand(*line*, *exclusion_string*, *global_ports*, *out_dir*)
Takes input from the BusinessUnit object that owns this ScanObject and creates the command for the system to execute.

GetMachineCount()
Determines the number of IP's that will be scanned with this object's instance.

Emailing Module

Purpose of Emailing

The Eamiling module send push notifications to white listed email handles specified in the config files.

Using Emailing

Emailing is a module specifically called by the user. No other module explicitly or implicitly imports this set of functions, but does need a BusinessUnit Object to format and send emails.

```
from portscan import Emailing

...

# After running a BusinessUnit Scan
BU.Collect()

if BU.emails > 0:
  Emailing.SendMail(BU)
```

Emailing methods

HTMLGenerator Module

Purpose of HTMLGenerator

The HTMLGenerator module provides a programtic generator for HTML output using data provided by a BusinessUnit object. The current version is fixed, and generates only once version of output.

Using HTMLGenerator

The BusinessUnit object implicitly calls functions inside of HMTLGenerator, but functions are also avalible to users. GenerateHTML requires a reference to a BusinessUnit object to collect data to generate the HTML report. An VERY basic example is provided below.

```
from portscan imoport HTMLGenerator

...

BU.Collect()

GenerateHTML(BU)
```

HTMLGenerator methods

Log Module

Purpose of Log

Log to a message a record it in a specific format created inside of the module.

Using Log

Log is a module for recording timestamped reports into a designated hidden log file. Logs can be viewd in real time by viewing the file:

```
$ {editor} .log
```

The function provided by this module is very simple and only requires a messgae to be appended to the default form being logged.

```
from portscan import Log

send_log("This is a test log")

$ cat .log
>> INFO:root:2017-09-07 01:44:40 This is a test log
```

Log methods

Upload Module

Purpose of Upload

This module allows uploading to DropBox to keep reports in a remote accessible repository.

Using Upload

This module is implicitly called the BusinessUnit object, but can still be called by the user. It requires both a Google URL Shortnerer api key defined in an environment variable named 'google_key', and a DropBox API key defined in an environment variable named 'dropbox_key'. If these are not defined a EnvironmentException will be raised and caught in the BusinessUnit Object, and no files will be uploaded.

```
$ export google_key='{YOUR GOOGLE API KEY}'
$ export dropbox_key='{YOUR DROPBOX API KEY}'
```

The function provided by the Upload module requires a list of files to be uploaded and a relative path inside of DropBox to upload to.

```
from portscan import Upload

...

#relative filepath inside of DropBox
path = "/nmap-test/"
```

```
files = ['out.html','test.html']

UploadToDropbox(files, path)
```

Upload methods

Server Requirements

Email

The emailing module will by deafult use the localhost SMTP server, but can take an argument to direct to another. Therefore use your preferred SMTP server, and configure it to send configured emails from python3.

PostFix is a recommended easy to use *nix server: http://www.postfix.org

Python Version

This project uses Python 3.x.x, and is not backward compatible.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

Symbols

C

G

P

S