Margarita Shotgun Documentation

Release 1.0

Joel Ferrier

September 16, 2016

Contents

Python Remote Memory Aquisition

CHAPTER 1

Quick Start

First, Install margaritashotgun.

1.1 Capture A Single Machine

A single machine can be captured using only the command line arguments for margaritashotgun. First specify the server and user with the `-s` and `-u` flags respectively. Next provide a path to an ssh key with `-k` (or use a password with the `-p` flag). Finally provide a lime kernel module with `-m` and specify an output file with `-f`

```
margaritashotgun -s 172.16.20.10 -u root -k root_access.pem -m lime-3.13.0-74-generic.ko -f 172.16.20
```

1.2 Save Memory In S3

To save a file to s3 simply replace the `-f` or `filename` flags with `-b` or `--bucket`. Ensure that you have aws credentials configured prior to executing the following command.

```
margaritashotgun -s 172.16.20.10 -u root -k root_access.pem -m lime-3.13.0-74-generic.ko -b memory_ca
```

1.3 Capture Multiple Machines

Run margaritashotgun with a configuration file like `parallel_config.yml.example`

```
aws:
    bucket: memory_dump_example
hosts:
    - addr:     52.36.191.XXX
      port:     22
      username: ec2-user
      key:      access.pem
      module:   lime-4.1.19-24.31.amzn1.x86_64.ko
    - addr:     52.36.170.XXX
      port:     22
      username: ec2-user
      key:      access.pem
      module:   lime-4.1.19-24.31.amzn1.x86_64.ko
    - addr:     52.36.210.XXX
      port:     22
```

```
      username: ubuntu
      key:      dev.pem
      module:   lime-3.13.0-74-generic.ko
    - addr:     52.36.90.XXX
      port:     22
      username: ubuntu
      key:      dev.pem
      module:   lime-3.13.0-74-generic.ko
workers: 2
```

Here parallelism is limited to 2 workers.

Run the capture with:

```
margaritashotgun -c your_custom_config.yml.
```

CHAPTER 2

Installation

2.1 System Requirements

Currently only linux is a supported platform. Running on OSX or Windows may be possible with minor modifications. While margaritashotgun is written purely in python, some of the libraries used require additional system packages.

2.1.1 Fedora / RHEL Distributions

- python-devel (2.X or 3.X)
- python-pip
- libffi-devel
- openssl-devel

2.1.2 Debian Distributions

- python-dev (2.X or 3.X)
- python-pip
- libffi-dev
- libssl-dev

2.2 Install From PyPi

Margaritashotgun is not currently listed in PyPi, while we work on that install via one of the methods below.

2.3 Installing From Github

```
$ pip install git+ssh://git@github.com/ThreatResponse/margaritashotgun.git@master
$ margaritashotgun -h
```

2.4 Local Build and Install

```
$ git clone https://github.com/ThreatResponse/margaritashotgun.git
$ cd margaritashotgun
$ python setup.py
$ pip install dist/margarita_shotgun-*.tar.gz
$ margaritashotgun -h
```

2.5 Local Execution

In the previous two example dependencies are automatically resolved, if you simply want to run margaritashotgun using the script `bin/margaritashotgun` you will have to manually install dependencies

```
$ git clone https://github.com/ThreatResponse/margaritashotgun.git
$ cd margaritashotgun
$ pip install -r requirements.txt
$ ./bin/margaritashotgun -h
```

CHAPTER 3

User Guide

Contents

3.1 Command Line

3.1.1 Common Examples

See the quickstart for common examples.

3.1.2 Usage

`margaritashotgun` has man configuration flags which are outlined in detail below.

```
$ margaritashotgun -h
usage: margaritashotgun [-h] (-c CONFIG | -s SERVER) [-P PORT] [-u USERNAME]
                        [-m MODULE] [-p PASSWORD] [-k KEY] [-f FILENAME]
                        [--repository] [--repository-url REPOSITORY_URL]
                        [-w WORKERS] [-v] [-b BUCKET | -o OUTPUT_DIR]
                        [-d LOG_DIR] [--log_prefix LOG_PREFIX]

Remote memory aquisition wrapper for LiME

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG, --config CONFIG
                        path to config.yml
  -s SERVER, --server SERVER
                        hostname or ip of target server
  -b BUCKET, --bucket BUCKET
                        memory dump output bucket
  -o OUTPUT_DIR, --output_dir OUTPUT_DIR
                        memory dump output directory

  -P PORT, --port PORT  ssh port on remote server
  -u USERNAME, --username USERNAME
                        username for ssh connection
  -m MODULE, --module MODULE
                        path to kernel lime kernel module
  -p PASSWORD, --password PASSWORD
                        password for user or encrypted keyfile
  -k KEY, --key KEY     path to rsa key for ssh connection
  -f FILENAME, --filename FILENAME
                        memory dump filename
  --repository          enable automatic kernel module downloads
  --repository-url REPOSITORY_URL
                        repository url
  -w WORKERS, --workers WORKERS
                        number of workers to run in parallel,default: auto
                        acceptable values are(INTEGER | "auto")
  -v, --verbose         log debug messages

  -d LOG_DIR, --log_dir LOG_DIR
                        log directory
  --log_prefix LOG_PREFIX
                       log file prefix
```

3.1.3 Config

The `-c` and `--config` flags accept a relative or absolute path to a yml config file. The structure of this file is outlided in the `Configuration` section below.

3.1.4 Server

The `-s` and `--server` flags specify the server being targeted for memory capture. A DNS record or IP address are valid inputs.

3.1.5 Bucket

The `-b` and `--bucket` flags specify the destination bucket when dumping memory to s3. This flag cannot be used in conjunction wth `-o` or `--output_dir`.

3.1.6 Output_Dir

The `-o` and `--output_dir` flags specify the destination folder when dumping memory to the local filesystem. This flag cannot be used in conjunction with `-b` or `--bucket`.

3.1.7 Port

The `-p` and `--port` flags specify the port that ssh is running on the remote server specified by `-s` or `--server`. This flag is optional and port `22` will be assumed if no value is provided.

3.1.8 Username

The `-u` and `--username` flags specify the user account to authenticate with when connecting to the remote server specified by `-s` or `--server`.

3.1.9 Module

The `-m` and `--module` flags accept a relative or absolute path to a LiME kernel module. This flag is required if no kernel module repository is enabled with the `--repository` flag.

3.1.10 Password

The `-p` and `--password` flags specify the password used for authentication with connection to the remote server specified by `-s` or `--server`. When used in conjuction with the `-k` or `--key` flags this password will be used to unlock a protected private key file.

3.1.11 Key

The `-k` and `--key` flags accept a relative or absolute path to a a private key file used for authentication when connecting to the server specified by `-s` or `-server`. If the private key file specified is password protected use the `-p` or `--password` flags to specify the password that unlocks the private key.

3.1.12 Filename

The `-f` and `--filename` flags specify the name of the file memory will be saved to when dumping to the local filesystem. The file will be saved to the local directory unless the `-o` or `--output_dir` options are configured.

3.1.13 Repository

The `--repository` flag enables automatic kernel module resolution via the repository configured with `--repository-url`. Margaritashotgun will not query any repositories unless explicitly enabled with the `--repository` flag.

3.1.14 Repository_Url

The `--repository-url` flag specifies where to search for kernel modules. The default public repository provided by Threat Response is availible at `https://threatresponse-lime-modules.s3.amazonaws.com`

3.1.15 Workers

The `-w` and `--workers` flags specify how many worker processes will be spawned to process memory captures in parallel. The default value for this flag is `auto` which will spawn a process per remote host up to the number of cpu cores on the local system. Integer values can be provided instead of the `auto` keyword. Eg. `--workers 3` will process 3 memory captures simultaneously.

3.1.16 Verbose

The `-v` and `--verbose` flags enable debug logging, including each command executed on remote hosts as a part of the memory capture process.

3.1.17 Log_Dir

The `-d` and `--log_dir` flags specify the directory in which to log files will be saved during memory capture.

3.1.18 Log_Prefix

The `--log_prefix` flag allows a custom case number to be prepended onto log files for easy identification.

3.2 Configuration File

Example configuration files are availible in the repository. More documentation about the configuration file format is in the works.

3.3 Managing AWS Credentials

Margaritashotgun does not support explicitly declaring aws credentials. Currently the only way to interact with S3 is by configuring an aws profile. A feature is planned to allow selecting a profile other than the `default` profile. Until that feature is completed the `default` profile must be used.

3.4 Wrapping Margarita Shotgun

Margarita Shotgun can be driven by another program when included as a python module. The configuration object passed to the margaritashotgun client must have the exact structure of the configuration file outlined above.

3.4.1 Example

```
>>> import margaritashotgun
>>> config = dict(aws dict(bucket = 'case-bucket'),
...               hosts = [ dict(addr = '10.10.12.10',
...                              port = 22,
...                              username = 'ec2-user',
...                              key = '/path/to/private-key') ]
...               workers = 'auto',
...               logging = dict(log_dir = 'logs/',
...                              prefix = 'casenumber-10.10.12.10'),
...               repository = dict(enabled = true,
...                                 url = 'your-custom-kernel-module-repo.io'))
...
>>> capture_client = margaritashotgun.client(name='mem-capture', config=config,
...                                          library=True, verbose=False)
...
>>> response = capture_client.run()
>>> print(response)
{'total':1,'failed':[],'completed':['10.10.12.10']}
```

Note that calling `capture_client.run()` is a blocking operation.

3.4.2 Real world implementation

An example of wrapping margaritashotgun is the project aws ir availible on github.

CHAPTER 4

Reference Guide

4.1 Authentication

class margaritashotgun.auth.**Auth**(*username=None*, *password=None*, *key=None*)

__init__(*username=None*, *password=None*, *key=None*)

Parameters

- **username** (*str*) – username for ssh authentication
- **password** (*str*) – password for ssh authentication
- **key** (*str*) – path to rsa key for ssh authentication

__module__ = 'margaritashotgun.auth'

load_key(*key_path*, *password*)
Creates paramiko rsa key

Parameters

- **key_path** (*str*) – path to rsa key
- **password** (*str*) – password to try if rsa key is encrypted

class margaritashotgun.auth.**AuthMethods**

__format__(*format_spec*)

__module__ = 'margaritashotgun.auth'

static __new__(*value*)

__reduce_ex__(*proto*)

__repr__()

__str__()

key = <AuthMethods.key: 'key'>

password = <AuthMethods.password: 'password'>

4.2 Client

class margaritashotgun.client.**Client**(*config=None*, *library=True*, *name=None*, *verbose=False*)
Client for parallel memory capture with LiME

__init__(*config=None*, *library=True*, *name=None*, *verbose=False*)

Parameters

- **library** (*bool*) – Toggle for command line features
- **config** (*dict*) – Client configuration

__module__ = 'margaritashotgun.client'

map_config()

run()
Captures remote hosts memory

statistics(*results*)

4.3 Cli

class margaritashotgun.cli.**Cli**

__module__ = 'margaritashotgun.cli'

check_directory_path(*path*)
Ensure directory exists at the provided path

Parameters **path** (*string*) – path to directory to check

check_directory_paths(**args*)
Ensure all arguments correspond to directories

check_file_path(*path*)
Ensure file exists at the provided path

Parameters **path** (*string*) – path to directory to check

check_file_paths(**args*)
Ensure all arguments provided correspond to a file

configure(*arguments=None*, *config=None*)
Merge command line arguments, config files, and default configs

Params arguments Arguments produced by Cli.parse_args

Params config configuration dict to merge and validate

configure_args(*arguments*)
Create configuration has from command line arguments

Params arguments arguments produced by *Cli.parse_args()*

get_env_default(*variable*, *default*)
Fetch environment variables, returning a default if not found

load_config(*path*)
Load configuration from yaml file

Parameters **path** (*string*) – path to configuration file

parse_args(*args*)
Parse arguments and return an arguments object

```
>>> from margaritashotgun.cli import Cli
>>> cli = CLi()
>>> cli.parse_args(sys.argv[1:])
```

Parameters **args** (*list*) – list of arguments

validate_config(*config*)
Validate configuration dict keys are supported

Parameters **config** (*dict*) – configuration dictionary

4.4 Exceptions

exception margaritashotgun.exceptions.**AuthenticationMethodMissingError**
Raised when no ssh authentication methods are specified

__init__()

__module__ = 'margaritashotgun.exceptions'

exception margaritashotgun.exceptions.**AuthenticationMissingUsernameError**
Raised when authentication method is configured without a username

__init__()

__module__ = 'margaritashotgun.exceptions'

exception margaritashotgun.exceptions.**InvalidConfigurationError**(*key*, *value*, *reason='unsupported configuration'*)
Raised when an unsupported configuration option is supplied

__init__(*key*, *value*, *reason='unsupported configuration'*)

__module__ = 'margaritashotgun.exceptions'

exception margaritashotgun.exceptions.**KernelModuleNotFoundError**(*kernel_version*, *repo_url*)
Raised when no kernel module is provided and a suitable module cannot be found

__init__(*kernel_version*, *repo_url*)

__module__ = 'margaritashotgun.exceptions'

exception margaritashotgun.exceptions.**KernelModuleNotProvidedError**(*kernel_version*)
Raised when no kernel module is provided and repository is disabled

__init__(*kernel_version*)

__module__ = 'margaritashotgun.exceptions'

exception margaritashotgun.exceptions.**LimeRetriesExceededError**(*retries*)
Raised when max number of retries are exceeded waiting for LiME to load.

__init__(*retries*)

__module__ = 'margaritashotgun.exceptions'

exception margaritashotgun.exceptions.**MargaritaShotgunError**
Base Error Class

__module__ = 'margaritashotgun.exceptions'

__weakref__
list of weak references to the object (if defined)

exception margaritashotgun.exceptions.**MemoryCaptureAttributeMissingError**(*attribute*)
Raised when memory capture is missing a required attribute

__init__(*attribute*)

__module__ = 'margaritashotgun.exceptions'

exception margaritashotgun.exceptions.**MemoryCaptureOutputMissingError**(*remote_host*)
Raised when no output is configured when capturing memory

__init__(*remote_host*)

__module__ = 'margaritashotgun.exceptions'

exception margaritashotgun.exceptions.**NoConfigurationError**
Raised when no configuration is supplied while operating as a library

__init__()

__module__ = 'margaritashotgun.exceptions'

exception margaritashotgun.exceptions.**SSHConnectionError**(*host*, *inner_exception*)
Raised when paramiko is unable to connect to a remote host

__init__(*host*, *inner_exception*)

__module__ = 'margaritashotgun.exceptions'

4.5 Logging

class margaritashotgun.logger.**Logger**(**args*, ***kwargs*)

__init__(**args*, ***kwargs*)

__module__ = 'margaritashotgun.logger'

margaritashotgun.logger.**cleanup**(*log_file*)

margaritashotgun.logger.**get_times**()

margaritashotgun.logger.**listener**(*queue*, *name*, *log_file*, *desc*)

4.6 Memory

class margaritashotgun.memory.**Memory**(*remote_addr*, *mem_size*, *progressbar=False*, *recv_size=1048576*, *sock_timeout=1*)

__init__(*remote_addr*, *mem_size*, *progressbar=False*, *recv_size=1048576*, *sock_timeout=1*)

Parameters

- **remote_addr** (*str*) – hostname or ip address of target server

- **mem_size** (*int*) – target server memory size in bytes
- **progressbar** (*bool*) – ncurses progress bar toggle
- **recv_size** (*int*) – transfer socket max receive size
- **sock_timeout** (*int*) – transfer socket receive timeout

__module__ = 'margaritashotgun.memory'

capture(*tunnel_addr*, *tunnel_port*, *filename=None*, *bucket=None*, *destination=None*)
Captures memory based on the provided OutputDestination

Parameters

- **tunnel_port** (*int*) – ssh tunnel hostname or ip
- **tunnel_port** – ssh tunnel port
- **filename** (*str*) – memory dump output filename
- **bucket** (*str*) – output s3 bucket
- **destination** (*margaritashotgun.memory.OutputDestinations*) – OutputDestinations member

cleanup()
Release resources used during memory capture

max_size(*mem_size*, *padding_percentage*)
Calculates the excpected size in bytes of the memory capture

Parameters

- **mem_size** (*int*) – target server memory in bytes
- **padding_percentage** (*float*) – Output overhead of lime format

to_file(*filename*, *tunnel_addr*, *tunnel_port*)
Writes memory dump to a local file

Parameters

- **filename** (*str*) – memory dump output filename
- **tunnel_port** (*int*) – ssh tunnel hostname or ip
- **tunnel_port** – ssh tunnel port

to_s3(*bucket*, *filename*, *tunnel_addr*, *tunnel_port*)
Writes memory dump to s3 bucket

Parameters

- **bucket** (*str*) – memory dump output s3 bucket
- **filename** (*str*) – memory dump output filename
- **tunnel_port** (*int*) – ssh tunnel hostname or ip
- **tunnel_port** – ssh tunnel port

update_progress(*complete=False*)
Logs capture progress

Params complete toggle to finish ncurses progress bar

class margaritashotgun.memory.**OutputDestinations**

__format__ (*format_spec*)

__module__ = 'margaritashotgun.memory'

static __new__ (*value*)

__reduce_ex__ (*proto*)

__repr__()

__str__()

local = <OutputDestinations.local: 'local'>

s3 = <OutputDestinations.s3: 's3'>

4.7 Remote Host

class margaritashotgun.remote_host.**Host**

__init__()

__module__ = 'margaritashotgun.remote_host'

capture_memory (*destination*, *filename*, *bucket*, *progressbar*)

check_for_lime (*pattern*, *listen_port*)
Check to see if LiME has loaded on the remote system

Parameters

- **pattern** (*str*) – pattern to check output against
- **listen_port** (*int*) – port LiME is listening for connections on

cleanup()
Release resources used by supporting classes

connect (*username*, *password*, *key*, *address*, *port*)
Connect ssh tunnel and shell executor to remote host

Parameters

- **username** (*str*) – username for authentication
- **password** (*str*) – password for authentication, may be used to unlock rsa key
- **key** (*str*) – path to rsa key for authentication
- **address** (*str*) – address for remote host
- **port** (*int*) – ssh port for remote host

kernel_version()
Returns the kernel kernel version of the remote host

load_lime (*remote_path*, *listen_port*, *dump_format='lime'*)
Load LiME kernel module from remote filesystem

Parameters

- **remote_path** (*str*) – path to LiME kernel module on remote host
- **listen_port** (*int*) – port LiME uses to listen to remote connections

- **dump_format** (*str*) – LiME memory dump file format

log_async_result(*future*)

mem_size()
Returns the memory size in bytes of the remote host

start_tunnel(*local_port*, *remote_address*, *remote_port*)
Start ssh forward tunnel

Parameters

- **local_port** (*int*) – local port binding for ssh tunnel
- **remote_address** (*str*) – remote tunnel endpoint bind address
- **remote_port** (*int*) – remote tunnel endpoint bind port

unload_lime()
Remove LiME kernel module from remote host

upload_module(*local_path=None*, *remote_path='/tmp/lime.ko'*)
Upload LiME kernel module to remote host

Parameters

- **local_path** (*str*) – local path to lime kernel module
- **remote_path** (*str*) – remote path to upload lime kernel module

wait_for_lime(*listen_port*, *listen_address='0.0.0.0'*, *max_tries=20*, *wait=1*)
Wait for lime to load unless max_retries is exceeded

Parameters

- **listen_port** (*int*) – port LiME is listening for connections on
- **listen_address** (*str*) – address LiME is listening for connections on
- **max_tries** (*int*) – maximum number of checks that LiME has loaded
- **wait** (*int*) – time to wait between checks

margaritashotgun.remote_host.**process**(*conf*)

4.8 Remote Shell

class margaritashotgun.remote_shell.**Commands**

__format__(*format_spec*)

__module__ = 'margaritashotgun.remote_shell'

static __new__(*value*)

__reduce_ex__(*proto*)

__repr__()

__str__()

kernel_version = <Commands.kernel_version: 'uname -r'>

lime_check = <Commands.lime_check: 'netstat -lnt | grep {0}'>

lime_pattern = <Commands.lime_pattern: '{0}:{1}'>

load_lime = <Commands.load_lime: 'sudo insmod {0} "path=tcp:{1}" format={2}'>

mem_size = <Commands.mem_size: "cat /proc/meminfo | grep MemTotal | awk '{ print $2 }'">

unload_lime = <Commands.unload_lime: 'sudo pkill insmod; sudo rmmod lime'>

class margaritashotgun.remote_shell.**RemoteShell**(*max_async_threads=2*)

__init__(*max_async_threads=2*)

Parameters **args** (*int*) – maximun number of async command executors

__module__ = 'margaritashotgun.remote_shell'

cleanup()
Release resources used during shell execution

connect(*auth*, *address*, *port*)
Creates an ssh session to a remote host

Parameters

- **auth** (*margaritashotgun.auth.AuthMethods*) – Authentication object
- **address** (*str*) – remote server address
- **port** (*int*) – remote server port

connect_with_key(*username*, *key*, *address*, *port*)
Create an ssh session to a remote host with a username and rsa key

Parameters

- **username** (*str*) – username used for ssh authentication
- **key** (paramiko.key.RSAKey) – paramiko rsa key used for ssh authentication
- **address** (*str*) – remote server address
- **port** (*int*) – remote server port

connect_with_password(*username*, *password*, *address*, *port*)
Create an ssh session to a remote host with a username and password

Parameters

- **username** (*str*) – username used for ssh authentication
- **password** (*str*) – password used for ssh authentication
- **address** (*str*) – remote server address
- **port** (*int*) – remote server port

decode(*stream*, *encoding='utf-8'*)
Convert paramiko stream into a string

Parameters

- **stream** – stream to convert
- **encoding** (*str*) – stream encoding

execute(*command*)
Executes command on remote hosts

Parameters **command** (*str*) – command to be run on remote host

execute_async(*command*, *callback=None*)
Executes command on remote hosts without blocking

Parameters

- **command** (*str*) – command to be run on remote host
- **callback** (*function*) – function to call when execution completes

upload_file(*local_path*, *remote_path*)
Upload a file from the local filesystem to the remote host

Parameters

- **local_path** (*str*) – path of local file to upload
- **remote_path** (*str*) – destination path of upload on remote host

4.9 Repository

class margaritashotgun.repository.**Repository**(*url*)

__init__(*url*)

__module__ = 'margaritashotgun.repository'

fetch_module(*urn*, *filename=None*, *chunk_size=1024*, *verify=False*)

list_modules()

search_modules(*kernel_version*)

verify_module_signature()

4.10 SSH Tunnel

class margaritashotgun.ssh_tunnel.**Forward**(*local_port*, *remote_address*, *remote_port*, *transport*)

__init__(*local_port*, *remote_address*, *remote_port*, *transport*)
type: local_port: int param: local_port: local tunnel endpoint ip binding type: remote_address: str param: remote_address: Remote tunnel endpoing ip binding type: remote_port: int param: remote_port: Remote tunnel endpoint port binding type: transport: paramiko.Transport param: transport: Paramiko ssh transport

__module__ = 'margaritashotgun.ssh_tunnel'

forward_tunnel(*local_port*, *remote_address*, *remote_port*, *transport*)

run()

stop()

class margaritashotgun.ssh_tunnel.**ForwardServer**(*server_address*, *RequestHandlerClass*, *bind_and_activate=True*)

__module__ = 'margaritashotgun.ssh_tunnel'

allow_reuse_address = True

daemon_threads = True

class margaritashotgun.ssh_tunnel.**Handler**(*request*, *client_address*, *server*)

__module__ = 'margaritashotgun.ssh_tunnel'

handle()

class margaritashotgun.ssh_tunnel.**SSHTunnel**

__init__()

__module__ = 'margaritashotgun.ssh_tunnel'

cleanup()
Cleanup resources used during execution

connect(*auth*, *address*, *port*, *hostkey=None*)
Connect paramico transport

Parameters

- **auth** (*:py:class 'margaritashotgun.auth.AuthMethods '*) – authentication object
- **address** (*str*) – remote server ip or hostname
- **port** (*int*) – remote server port
- **hostkey** (paramiko.key.HostKey) – remote host ssh server key

connect_with_key(*username*, *key*, *hostkey=None*)
Connect paramico transport with public key authentication

Parameters

- **username** (*str*) – ssh authentication username
- **key** (paramiko.key.RSAKey) – ssh authentication private key
- **hostkey** (paramiko.key.HostKey) – remote host ssh server key

connect_with_password(*username*, *password*, *hostkey=None*)
Connect paramico transport with password authentication

Parameters

- **username** (*str*) – ssh authentication username
- **password** (*str*) – ssh authentication password
- **hostkey** (paramiko.key.HostKey) – remote host ssh server key

start(*local_port*, *remote_address*, *remote_port*)
Start ssh tunnel

type: local_port: int param: local_port: local tunnel endpoint ip binding type: remote_address: str param: remote_address: Remote tunnel endpoing ip binding type: remote_port: int param: remote_port: Remote tunnel endpoint port binding

4.11 Workers

class `margaritashotgun.workers.`**`Workers`**(*conf*, *workers*, *name*, *library=True*)

`__init__`(*conf*, *workers*, *name*, *library=True*)

`__module__` = **'margaritashotgun.workers'**

`cleanup`(*terminate=False*)

`count`(*workers*, *cpu_count*, *host_count*)

`cpu_count` = **None**

`hosts` = **None**

`progress_bar` = **True**

`spawn`(*desc*, *timeout=1800*)

`worker_count` = **None**

CHAPTER 5

Architecture

An Overview of margaritashotugn's architecture Coming Soon!

CHAPTER 6

Development

6.1 Tests

The test suite is written with pytest and can be run with `py.test --cov=margaritashotgun`

CHAPTER 7

About

Margaritashotgun is a part of the Threat Response project.

7.1 License

Margarita Shotgun is distributed under the MIT License (MIT).

m

Symbols

A

C

D

E

F

G

H

I

K

L

M