Margarita Shotgun Documentation

Release 0.3.1

Joel Ferrier

February 21, 2017

Contents

Python Remote Memory Aquisition

CHAPTER 1

Quick Start

First, Install margaritashotgun.

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 `--server` and `--username` flags. Next provide a path to an ssh key with `--key` (or use a password with the `--password` flag). Finally provide a lime kernel module with `--module` and specify an output file with `--filename`

```
margaritashotgun --server 172.16.20.10 --username root --key root_access.pem --module lime-3.13.0-74-
```

Save Memory In S3

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

```
margaritashotgun --server 172.16.20.10 --username root --key root_access.pem --module lime-3.13.0-74-
```

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
```

Note: In this example parallelism is limited to 2 workers.

Run the capture with:

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

CHAPTER 2

Installation

System Requirements

Margarita Shotgun is supported on common linux distributions, for other operating systems see the Install with Docker section.

While margaritashotgun is written purely in python, some of the libraries used require additional system packages.

Fedora / RHEL Distributions

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

Debian Distributions

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

Install From PyPi

```
$ pip install margaritashotgun
```

Install with Docker

Pull and run the python docker image.

```
$ docker pull python:3
$ docker run -ti python:3 bash
$ root@3009a5bc9817:/# pip install margaritashotgun
```

Note: If you plan on streaming memory to S3 ensure you setup IAM access keys in the docker container. Set `-e ACCESS_KEY_ID=ACCESS_KEY_ID -e AWS_SECRET_ACCESS_KEY=ACCESS_KEY` in the `docker run` command above. Alternately follow Amazon's guide for configuring credentials once the docker container is running.

Install From Github

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

Local Build and Install

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

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

Command Line

Note: See the quickstart for common examples.

Usage

Run `margaritashotgun -h` at the command line, detailed information on flags is below.

Quick Reference

Flag	Use	Notes
`--config`	path to config file	See the **Configuration File** section
`--server`	ip of remote server	DNS records may also be used
`--version`	print version info	
`--bucket`	output S3 bucket	Incommpatible with `-o`
`--output-dir`	local output folder	Incompatible with `-b`
`--port`	ssh port	`22` is used unless specified
`--username`	ssh username	Username for ssh authentication
`--module`	lime kernel module	Required if no repository is enabled
`--password`	ssh password	Unlockes RSA key when used with `-k`
`--key`	RSA Key	Unlocked via `-p` if supplied
`--jump-server`	ip of jump host	DNS records may also be used
`--jump-port`	jump host ssh port	`22` is used unless specified
`--jump-username`	jump host ssh username	Username for jump host ssh authentication
`--jump-password`	jump host ssh password	
`--jump-key`	jump host RSA key	
`--filename`	output file	
`--repository`	enable kernel repo	Default state is disabled
`--repository-url`	custom repo url	Defaults to threat response modules
`--repository-manifest`	custom repo url	Defaults to "primary"
`--gpg-no-verify`	disable signature checks	
`--workers`	worker count	Constrains parallel captures
`--verbose`	log debug messages	
`--log-dir`	log directory	
`--log-prefix`	log file prefix	

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.

Server

The `--server` flag specifies the server being targeted for memory capture. A DNS record or IP address are valid inputs.

Version

The `--version` flag prints the module version.

Bucket

The `--bucket` flag specifies the destination bucket when dumping memory to s3. This flag cannot be used in conjunction wth `-o` or `--output-dir`.

Output-Dir

The `--output-dir` flags specify the destination folder when dumping memory to the local filesystem. This flag cannot be used in conjunction with `--bucket`.

Port

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

Username

The `--username` flag specifies the user account to authenticate with when connecting to the remote server specified by `--server`.

Module

The `--module` flag accepts 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.

Password

The `--password` flag specifies the password used for authentication with connection to the remote server specified by `--server`. When used in conjuction with the `--key` flag this password will be used to unlock a password protected private key file.

Key

The `--key` flag accepts a relative or absolute path to a a private key file used for authentication when connecting to the server specified by `-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.

Filename

The `--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 `--output-dir` option is configured.

Repository

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

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`

Repository-manifest

The `--repository-manifest` flag specifies alternate kernel module manifests in the remote repository configured by `--repository-url`. For more information on repository structure and manifests see the architecture page or lime-compiler repository.

Gpg-no-verify

The `--gpg-no-verify` flag disables gpg verification of kernel modules downloaded from a remote repository.

Workers

The `--workers` flag specifies 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.

Verbose

The `--verbose` flag enables debug logging, including each command executed on remote hosts as a part of the memory capture process.

Log-Dir

The `--log-dir` flag specifies the directory in which log files will be saved during memory capture.

Log-Prefix

The `--log-prefix` flag specifies a custom case number that is prepended onto log files.

Configuration File

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

Managing AWS Credentials

Margarita Shotgun 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.

Recommended IAM Policy

Margarita Shotgun only requires PutObject on a specified bucket.

Example

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::member-berries/*"
        }
    ]
}
```

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.

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.

Real world implementation

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

CHAPTER 4

Reference Guide

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'>

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*)

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

merge_config(*base*, *config*)

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

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.**ConfigurationMergeError**(*reason*)
Raised when merging user configuration with the base config fails

__init__(*reason*)

__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.**RepositoryError**(*metadata_url*, *reason*)
Raised when malformed repository metadata is found

__init__(*metadata_url*, *reason*)

__module__ = 'margaritashotgun.exceptions'

exception margaritashotgun.exceptions.**RepositoryMissingSignatureError**(*signature_url*)
Raised when a detached signature is missing in remote repository"

__init__(*signature_url*)

__module__ = 'margaritashotgun.exceptions'

exception margaritashotgun.exceptions.**RepositoryMissingSigningKeyError**(*url*)
Raised when signing public key is missing from repository

__init__(*url*)

__module__ = 'margaritashotgun.exceptions'

exception margaritashotgun.exceptions.**RepositorySignatureError**(*url*, *signature_url*)
Raised when signature verification fails

__init__(*url*, *signature_url*)

__module__ = 'margaritashotgun.exceptions'

exception margaritashotgun.exceptions.**RepositoryUntrustedSigningKeyError**(*url*, *fingerprint*)

Raised when repository signing key is not trusted

__init__(*url*, *fingerprint*)

__module__ = 'margaritashotgun.exceptions'

exception margaritashotgun.exceptions.**SSHCommandError**(*host*, *command*, *message*)

Raised when an exception is encountered executing a command on a remote host

__init__(*host*, *command*, *message*)

__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'

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*)

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'>

Remote Host

class margaritashotgun.remote_host.**Host**

__init__()

__module__ = 'margaritashotgun.remote_host'

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

check_for_lime(*pattern*)
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*, *jump_host*)
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*)

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: 'cat /proc/net/tcp'>

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*, *jump_host*, *jump_auth*)
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_auth(*ssh*, *auth*, *address*, *port*, *sock*)

connect_with_key(*ssh*, *username*, *key*, *address*, *port*, *sock*, *timeout=20*)
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(*ssh*, *username*, *password*, *address*, *port*, *sock*, *timeout=20*)
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

transport()

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

Repository

class margaritashotgun.repository.**Repository**(*url*, *gpg_verify*)
Lime-compiler repository client https://github.com/ThreatResponse/lime-compiler

__init__(*url*, *gpg_verify*)

Parameters

- **url** (*str*) – repository url
- **gpg_verify** (*bool*) – enable/disable gpg signature verification

__module__ = 'margaritashotgun.repository'

check_signing_key()
Check that repo signing key is trusted by gpg keychain

fetch(*kernel_version*, *manifest_type*)
Search repository for kernel module matching kernel_version

Parameters

- **kernel_version** (*str*) – kernel version to search repository on
- **manifest_type** (*str*) – kernel module manifest to search on

fetch_module(*module*)
Download and verify kernel module

Parameters **module** (*str*) – kernel module path

get_manifest(*metadata*)
Get latest manifest as specified in repomd.xml

Parameters **metadata** (*dict*) – dictionary representation of repomd.xml

get_metadata()
Fetch repository repomd.xml file

get_signing_key()
Download a local copy of repo signing key and generate key metadata

init_gpg()
Initialize gpg object and check if repository signing key is trusted

install_key(*key_data*)
Install untrusted repo signing key

parse_manifest(*manifest_xml*)
Parse manifest xml file

Parameters **manifest_xml** (*str*) – raw xml content of manifest file

parse_metadata(*metadata_xml*)
Parse repomd.xml file

Parameters **metadata_xml** (*str*) – raw xml representation of repomd.xml

prompt_for_install()
Prompt user to install untrusted repo signing key

unzip_manifest(*raw_manifest*)
Decompress gzip encoded manifest

Parameters **raw_manifest** (*str*) – compressed gzip manifest file content

verify_checksum(*data*, *checksum*, *filename*)
Verify sha256 checksum vs calculated checksum

Parameters

- **data** (*str*) – data used to calculate checksum
- **checksum** (*str*) – expected checksum of data
- **checksum** – original filename

verify_data_signature(*signature_url*, *data_url*, *data*)
Verify data against it's remote signature

Parameters

- **signature_url** (*str*) – remote path to signature for data_url
- **data_url** (*str*) – url from which data was fetched
- **data** (*str*) – content of remote file at file_url

verify_file_signature(*signature_url*, *file_url*, *filename*)
Verify a local file against it's remote signature

Parameters

- **signature_url** (*str*) – remote path to signature for file_url
- **file_url** (*str*) – url from which file at filename was fetched
- **filename** (*str*) – filename of local file downloaded from file_url

verify_module(*filename*, *module*, *verify_signature*)
Verify kernel module checksum and signature

Parameters

- **filename** (*str*) – downloaded kernel module path
- **module** (*dict*) – kernel module metadata
- **verify_signature** (*bool*) – enable/disable signature verification

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

configure(*transport*, *auth*, *address*, *port*)
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

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

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

Multiprocessing Overview

Coming Soon.

Kernel Module Repository

Kernel Modules can be automatically downloaded from a remote repository with the `--repository` flag.

Repository Structure

Legacy repositories were dependent on the xml file listing provided by issuing a `GET` request to the root of an amazon s3 bucket.

Metadata files have been introduced to remove s3 hosting as a requirement for repositories. The official Threat Response repository will continue to be hosted in s3

Note: Kernel modules will continue to be availible at the root of the Threat Response repository until at least the `0.4.0` release of Margarita Shotgun. GPG Signatures will not be provided for these modules.

The new repository structure introduces several new requirements.

1. **Optional** The public portion of the GPG used for signing modules and metadata should present at the root of the repository with the filename `REPO_SIGNING_KEY.asc`. If the signing key is not present the `--gpg-no-verify` flag must be used with Margarita Shotgun.
2. **Required** A folder must exist in at the path `/repodata` which contains the following files.
   (a) **Required** `repomd.xml` contains repository metadata including one or more manifests of kernel modules.
   (b) **Optional** `repomd.xml.sig` detached signature for `repomd.xml`. If not present in the repository the `--gpg-no-verify` flag must be used with Margarita Shotgun.
   (c) **Optional** Manifest files. Techinally manifests can be stored at any relative path but it is recommended that they be stored in the `repodata` directory.
3. **Optional** A `modules` directory is recommended which will contain the following files. Note the following files can have any location relative to the repository root, the `modules` directory is simply best practice.

(a) **Required** Compiled lime kernel modules. Module filenames are arbitrary as the files are explicitly listed in a manifest

(b) **Optional** Detached kernel module signatures. If signatures are not present the `--gpg-no-verify` flag must be used with Margarita Shotgun. The signature filename is arbitary as it is explicitly listed in a manifest.

Below is an example directory listing of the repository structure.

```
-- REPO_SIGNING_KEY.asc
-- modules
|   -- lime-2.6.32-131.0.15.el6.centos.plus.x86_64.ko
|   -- lime-2.6.32-131.0.15.el6.centos.plus.x86_64.ko.sig
|   -- ...
|   -- lime-4.4.8-20.46.amzn1.x86_64.ko
|   -- lime-4.4.8-20.46.amzn1.x86_64.ko.sig
-- repodata
    -- a134928b6436bae3a9d9d1ddc47cb3c1539d4b559b266c84012ecb2e296b05a5-primary.xml.gz
    -- repomd.xml.sig
    -- repomd.xml
```

repomd.xml

The `repomd.xml` file contains repository metadata required to resolve kernel modules.

The `<revision></revision>` element contains the unix timestamp at which the metadata file was generated.

The structure of a `<data/>` element is described in the following table.

type	metadata file type
checksum	checksum of gzipped manifest file
open_checksum	checksum of deflated manifest file
location	href: path to gzipped manifest file
timestamp	last modified timestamp of deflated file
size	gzipped filesize
open_size	deflated filesize

Below is an example repository metadata listing.

```
<?xml version="1.0" encoding="UTF-8"?>
<metadata>
  <revision>1474605823</revision>
  <data type="primary">
    <checksum>8d9afbddd9041ed66d3da5db9b1d2d11fbdb5cf6a309d962ac4d1f66fb800551</checksum>
    <open_checksum>a134928b6436bae3a9d9d1ddc47cb3c1539d4b559b266c84012ecb2e296b05a5</open_checksum>
    <location href="repodata/a134928b6436bae3a9d9d1ddc47cb3c1539d4b559b266c84012ecb2e296b05a5-primary
    <timestamp>1474605823</timestamp>
    <size>20726</size>
    <open_size>143916</open_size>
  </data>
</metadata>
```

Module manifest

A manifest consists of multiple module elements inside of a modules element. The module tag and it's child elements are documented in the following table.

type	kernel module type
name	kernel module friendly name
arch	kernel module architecture
checksum	kernel module checksum
version	kernel version targeted by module
packager	packager of kernel module
location	href: path to module
signature	href: path to module signature
platform	the operating system this module targets

Below is a truncated manifest.

```
<?xml version="1.0" encoding="UTF-8"?>
<modules>
  <module type="lime">
    <name>lime-2.6.32-358.11.1.el6.x86_64.ko</name>
    <arch>x86_64</arch>
    <checksum>1d7fc899a95b050a4f434c07012279e84bdd95234420648fbf348f5b4289e9e6</checksum>
    <version>2.6.32-358.11.1.el6.x86_64</version>
    <packager>lime-compiler info@threatresponse.cloud</packager>
    <location href="modules/lime-2.6.32-358.11.1.el6.x86_64.ko"/>
    <signature href="modules/lime-2.6.32-358.11.1.el6.x86_64.ko.sig"/>
    <platform>linux</platform>
  </module>
  ...
  <module type="lime">
    <name>lime-3.10.0-327.28.2.el7.x86_64.ko</name>
    <arch>x86_64</arch>
    <checksum>203e04dbe23ffb0c59d41760e7e8ebc55117e270de6ee17e149107345be6ed0d</checksum>
    <version>3.10.0-327.28.2.el7.x86_64</version>
    <packager>lime-compiler info@threatresponse.cloud</packager>
    <location href="modules/lime-3.10.0-327.28.2.el7.x86_64.ko"/>
    <signature href="modules/lime-3.10.0-327.28.2.el7.x86_64.ko.sig"/>
    <platform>linux</platform>
  </module>
</modules>
```

GPG Signatures

Unless explicitly disabled all kernel modules and metadata files will be checked agains their gpg signature in remote repositories. Failure to verify a signature, or lack of a signature for a given file is considered a fatal error and will result in a failed memory capture.

Note: Disable signature verification with `--gpg-no-verify`. Checksum verification cannot be disabled.

Build Kernel Modules

Kernel modules are build and signed by the lime-compiler. The source is availible and will soon be distributed as a ruby gem for use building private repositories.

CHAPTER 6

Development

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.

License

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

m

Symbols

U

V

W