Hewlett Packard
Enterprise

Manual

Maltese

Malware Traffic Emulator Software

Requirements

- setuptools – library used to setup all dependencies for maltese
- dnspython3 – library used to create and resolve DNS requests
- scapy-python3 – library used to build and process network level packets.
- libdnet – a required library if using Mac OS X

Maltese is a tool that can be used to emulate malicious DNS traffic. This document details the various options that can be used to configure the tool.

The tool is written in python and uses the Python 3.4 environment. As of now, Maltese supports the use-case of emulating the DNS traffic of one type of malware on a given machine..

Architecture

Physical architecture

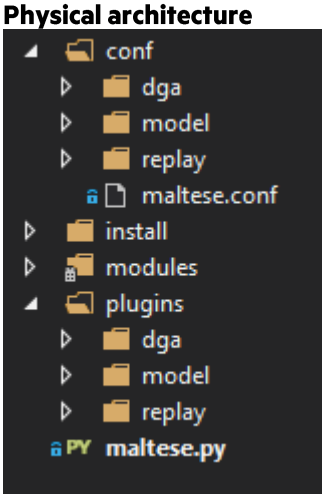
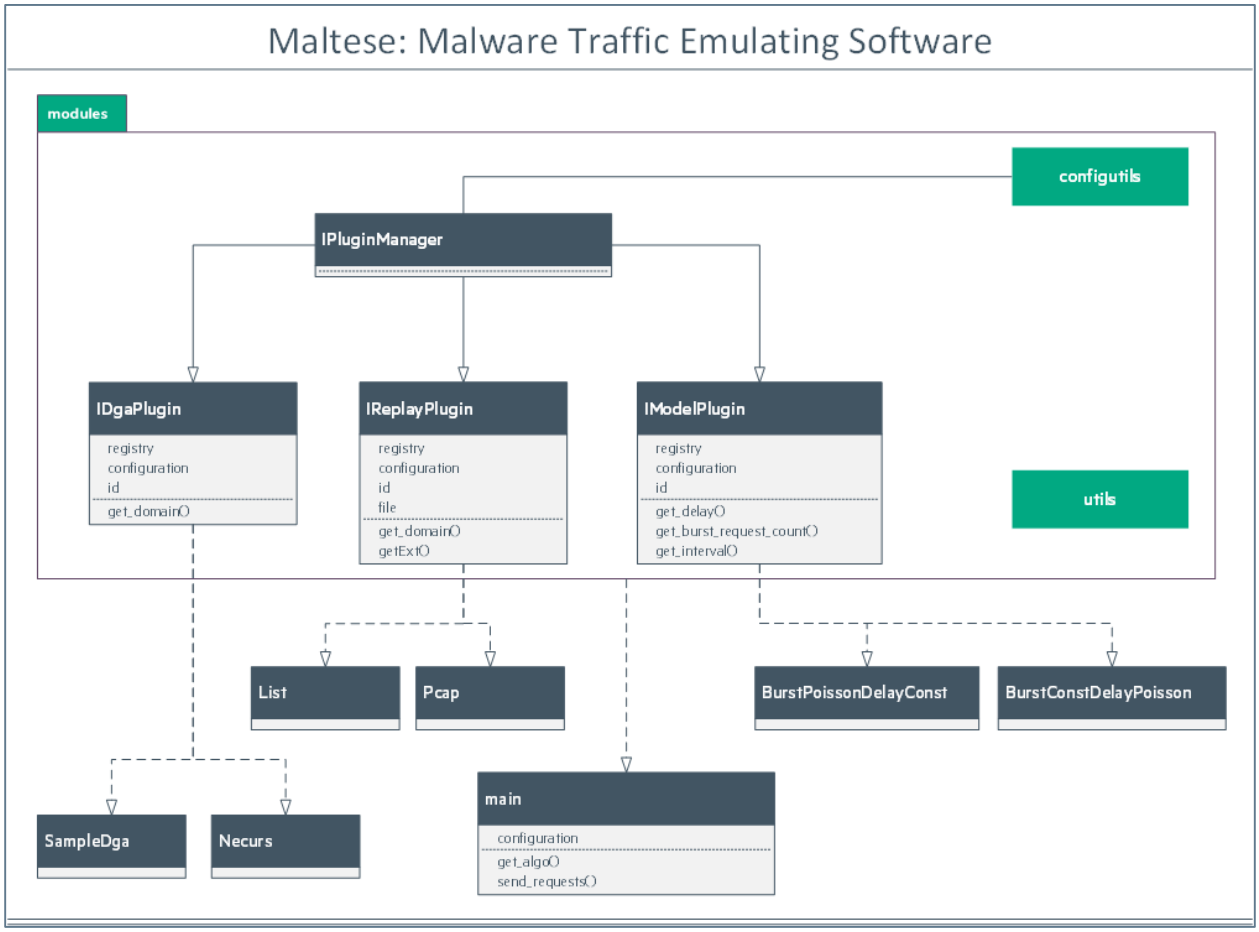


Figure 1. Maltese architecture

Tool execution occurs from the main file – Maltese.py. As seen in Figure 1, the architecture includes three main folders: conf – tool configurations, modules – the tool's core, and plugins – domain generation and model plugins.

Maltese is a plugin architecture. When newer plugins are added to the right folders the tool will automatically load and execute them based on the provided configuration.

Logical architecture



The tool core is comprised of the main and modules package. All other entities are plugins added to demonstrate Maltese's capabilities.

Getting Started

Install prerequisites

Python 3.4 is required. In order to install the Maltese's dependencies, go to the install folder and execute the following command.

```
python setup.py install
```

Syntax

The program can be started using the syntax

```
python maltese.py {dga,replay}
```

The main script takes one of two commands – `dga` or `replay`.

All commands accept the `-l` option which defines the logging level of the tool. This is set to the error level by default. The following levels are available.

- Error
- Warning
- Info
- Debug

The following options are available under `dga`.

- `-p`: The DGA plugin module

The following options are available under replay

- `-p`: The replay plugin module
- `-i`: The file (with path) to be replayed

The `-h` option can be used from any command to display a help menu with the available options.

Writing a plugin

Each plugin should be implemented in a separate `.py` file (python module) and placed under the appropriate folder. Some basic helpers are provided in the `modules.utils` module, but other modules may be imported as needed. It is recommended to declare a log object in the plugin and use it wherever required.

```
import logging
log = logging.getLogger(__name__)
```

All plugins have a predefined member variable named configuration. This is a dictionary and contains both the global and plugin specific configuration. If there is a conflict, the plugin configuration will always override the global configuration.

Note: Other than the default configuration options already available, each plugin can have its own custom configuration as required. The new configuration keys and values will automatically load into the plugin during execution.

Domain generation plugin

All `dga` and `replay` plugins are considered domain generation plugins. They are responsible for generating the requested domain names. The `get_domain()` function is the most important function to implement as it returns the generated domains.

It is important to note that the `get_domain()` function is called by a generator, which is a python feature allowing the function to generate multiple domains without using too much memory. The function may use loops to produce each domain. When a domain is produced, it should be returned using the yield operator in python.

```
yield domain
```

This sends the domain to the caller. Once the request is sent, program execution returns to the next statement following “yield”, preserving the previous program state, which continues looping to the next domain. This allows the tool to efficiently create a large number of strings by producing each domain on-demand, and discarding prior to the next one.

Model plugin

The model plugin describes the traffic pattern. Currently, three parameters are returned by any model:

- Burst count: The number of requests sent within a single burst of requests. This parameter is tested before every burst, allowing for burst size to vary.
- Interval: The time between requests in each burst. Given in seconds.
- Delay: The time between each burst of requests. Given in minutes.

Configuration

The global configuration offers the following options:

- Loop: If this option is set to true, once the plugin finishes generating all domains based on its logic, it will be called back to start generating again from the start. This allows the tool to generate domains for an infinite period (or until manually terminated)
- Dryrun: If this option is set to true, it allows Maltese to generate the domains printing them to the screen, but not sending any requests. This serves as a safe way to test the plugin’s algorithm before generating traffic. Setting it to false allows the DNS requests to be sent to the default name server.
- Model: This is the model that will be used to send requests generated by the domain generation plugin.
- Srcip: The source IP of the host may be spoofed to any given IP address.

Note that these configurations may also be provided at a plugin level if needed.

If a given configuration key has multiple values, each value should be separated by a space.

Note: The dryrun option is set to true by default. In order to send the generated traffic, it needs to be set to false.
