

# **Automated Malware and Exploit Data Collection Tool Documentation**

**version 0.3**

**Software Practicum Team 3**

April 04, 2020



# Contents

<b>Welcome to Automated Malware and Exploit Data Collection Tool's documentation!</b>	<b>1</b>
src	1
MainServer module	1
Entities package	1
Submodules	1
Entities.Entity module	1
Entities.ExploitInfo module	1
Entities.NetworkSettings module	2
Entities.Provision module	2
Entities.Response module	2
Entities.Scenario module	2
Entities.VagrantFile module	3
Entities.VirtualMachine module	3
Entities.VulnerabilityInfo module	3
Module contents	4
Managers package	4
Submodules	4
Managers.DatabaseManager module	4
Managers.FileManager module	4
Managers.ScenarioManager module	5
Managers.VagrantManager module	5
Module contents	6
<b>Indices and tables</b>	<b>6</b>
<b>Index</b>	<b>7</b>
<b>Python Module Index</b>	<b>9</b>



# Welcome to Automated Malware and Exploit Data Collection Tool's documentation!

## src

### MainServer module

`MainServer.createScenario (scenario_name)`

Creates a new scenario which includes the folders and the scenario JSON file :param scenario\_name: String with the scenario name :return: True if the new scenario was successfully created

`MainServer.deleteFile (file_name)`

`MainServer.deleteScenario (scenario_name)`

Edits a current scenario with a JSON file :param scenario\_name: String with the scenario name :return: True if the scenario has been successfully edited, otherwise False

`MainServer.editScenario ()`

Edits a current scenario with a JSON file :param scenario\_name: String with the scenario name :return: True if the scenario has been successfully edited, otherwise False

`MainServer.getAvailableBoxes ()`

Gets the available boxes in the Vagrant context :return: A list of string with the available boxes

`MainServer.getFileList ()`

`MainServer.getScenario (scenario_name)`

Gets the scenario as a JSON file :param scenario\_name: String with the scenario name :return: JSON file with the scenario info

`MainServer.getScenarios ()`

Gets the available scenarios :return: A list of strings with the available scenarios

`MainServer.runVagrantUp (scenario_name)`

Executes the vagrant up command for each machine in the scenario :param scenario\_name: String with the scenario name :return: True if the vagrant up commands were successfully executed

`MainServer.testPing (scenario_name, source, destination)`

Tests network connectivity between two virtual machines :param scenario\_name: String with the scenario name :param source: Source virtual machine :param destination: Destination virtual machine :return:

`MainServer.uploadFile ()`

### Entities package

#### Submodules

#### Entities.Entity module

`class Entities.Entity.Entity`

Bases: `abc.ABC`

**`abstract dictionary ()`**

**`abstract objectFromDictionary (dict)`**

#### Entities.ExploitInfo module

`class Entities.ExploitInfo.ExploitInfo (name='', type='', download_link='')`

Bases: **Entities.Entity.Entity**

**dictionary ()**

Generates a dictionary for the ExploitInfo object :return: A dictionary with ExploitInfo data

**objectFromDictionary (dict)**

### ***Entities.NetworkSettings module***

```
class Entities.NetworkSettings.NetworkSettings (network_name='', network_type='',  
ip_address='', auto_config=True)
```

Bases: **Entities.Entity.Entity**

**dictionary ()**

Generates a dictionary for the NetworkSettings object :return: A dictionary with NetworkSettings data

**objectFromDictionary (dict)**

### ***Entities.Provision module***

```
class Entities.Provision.Provision (name='', provision_type='shell')
```

Bases: **Entities.Entity.Entity**

**dictionary ()**

Generates a dictionary for the Provision object :return: A dictionary with Provision data

**objectFromDictionary (dict)**

**setShellCommand (command)**

Sets a new command to be executed as part of the provisioning :param command: Bash command intended for provisioning a virtual machine

### ***Entities.Response module***

```
class Entities.Response.Response (response='', code='', status='', task_id='', body='')
```

Bases: **Entities.Entity.Entity**

**dictionary ()**

Generates a dictionary for the Wrapper object :return: A dictionary with Wrapper data

**objectFromDictionary (dict)**

**setBody (body)**

**setCode (code)**

**setResponse (response)**

**setStatus (status)**

**setTask\_id (task\_id)**

### ***Entities.Scenario module***

```
class Entities.Scenario.Scenario (scenario_name)
```

Bases: **Entities.Entity.Entity**

#### **addVM (vm)**

Adds a new virtual machine to this scenario :param vm: Object which carries the virtual machine data

#### **dictionary ()**

Generates a dictionary for the Scenario object :return: A dictionary with Scenario data

#### **objectFromDictionary (dict)**

#### **setExploitInfo (exploit\_info)**

Sets the exploit info for this scenario :param exploit\_info: Object which carries the exploit info

#### **setVulnerabilityInfo (vulnerability\_info)**

Sets the vulnerability info for this scenario :param vulnerability\_info: Object which carries the vulnerability info

### ***Entities.VagrantFile module***

```
class Entities.VagrantFile.VagrantFile
```

Bases: **object**

#### **vagrantFilePerMachine (machine, machine\_path)**

Creates a vagrant file for this machine :param machine: Object which carries the virtual machine data :param machine\_path: Folder path to this machine :return: String used to write the vagrant file

### ***Entities.VirtualMachine module***

```
class Entities.VirtualMachine.VirtualMachine (name='', os='', is_attacker=False)
```

Bases: **Entities.Entity.Entity**

#### **addSharedFolder (hostPath, guestPath)**

Adds the shared folder between the host and the guest :param hostPath: String with the host path :param guestPath: String with the guest path

#### **dictionary ()**

Generates a dictionary for the Virtual Machine object :return: A dictionary with Virtual Machine data

#### **enableGUI (isVisible)**

Enables the GUI for this virtual machine :param isVisible: Boolean to enable or disable the GUI in a virtual machine

#### **objectFromDictionary (dict)**

#### **setName (name)**

Sets the name for this virtual machine :param name: String with the virtual machine name

#### **setNetworkSettings (network\_settings)**

Sets the network settings for this virtual machine :param network\_settings: Object which carries the network settings data

#### **setOS (os)**

Sets the OS for this virtual machine :param os: String with the virtual machine OS

#### **setProvision (provision)**

Sets the provision for this virtual machine :param provision: Object which carries the provision data

### ***Entities.VulnerabilityInfo module***

```
class Entities.VulnerabilityInfo.VulnerabilityInfo (name='', type='', cve_link='',
download_link='')
    Bases: Entities.Entity.Entity

    dictionary ()
        Generates a dictionary for the VulnerabilityInfo object :return: A dictionary with VulnerabilityInfo data

    objectFromDictionary (dict)
```

## Module contents

## Managers package

## Submodules

### Managers.DatabaseManager module

Run mongoDB server:

- cd C:\Program Files\MongoDB\Server\2.2\bin
- mongod

Create database:

- use soft\_prac

Create collections:

- db.createCollection('scenarios')

```
class Managers.DatabaseManager.DatabaseManager (url='')
    Bases: object

    deleteScenario (scenario_name)

    editScenario (scenario_json)

    getScenario (scenario_name)

    getScenarioNames ()

    getScenarios ()

    insertScenario (scenario_json)
```

### Managers.FileManager module

```
class Managers.FileManager.FileManager
    Bases: object

    createMachineFolders (scenario_json)
        Creates a folder for each machine in the scenario :param scenario_json: String with the scenario name :return:
        True if machine folders are created successfully

    createScenarioFolders (scenario_name)
        Creates a scenario folder with the JSON, Exploit, Vulnerability and Machines subfolders :param scenario_name:
        String with the scenario name :return: True if the scenario is created successfully

    getCurrentPath ()
```



Gets the project folder path :return: String with the project path

**getJSONPath (scenario\_name)**

**getScenariosPath ()**

Gets the scenarios folder path :return: String with the scenarios project path

## ***Managers.ScenarioManager module***

**class Managers.ScenarioManager.ScenarioManager (db\_manager='')**

Bases: **object**

**deleteScenario (scenario\_name)**

**editScenario (new\_scenario)**

Edits a current scenario with a JSON file :param scenario\_name: String with the scenario name :param scenario\_json: JSON file with the new scenario :return: True if the scenario has been successfully edited, otherwise False

**getScenario (scenario\_name)**

Gets the scenario as a JSON file :param scenario\_name: String with the scenario name :return: JSON file with the scenario info

**getScenarios ()**

Gets the available scenarios :return: A list of strings with the available scenarios

**newEmptyScenario (scenario\_name)**

Creates a new scenario which includes the folders and the scenario JSON file :param scenario\_name: String with the scenario name :return: True if the new scenario was successfully created

**scenarioExists (scenario\_name)**

Check if a scenario exists :param scenario\_name: String with the scenario name :return: False if the scenario JSON file does not exist and the path to the JSON file if it exist

**testDB (scenario\_name)**

**testDB2 ()**

## ***Managers.VagrantManager module***

**class Managers.VagrantManager.VagrantManager**

Bases: **object**

**createVagrantFiles (scenario\_name)**

Creates a vagrant file per machine in a scenario :param scenario\_json: String with the scenario name :return: True if vagrant files were successfully created

**getAvailableBoxes ()**

Gets the available boxes in the Vagrant context :return: A list of string with the available boxes

**haltVM (machine\_name)**

**restartVM (machine\_name)**

**runVagrantUp (scenario\_name)**

Executes the vagrant up command for each machine in the scenario :param scenario\_name: String with the scenario name :return: True if the vagrant up commands were successfully executed

```
sendCommand (scenario_name, machine_name, command, default_timeout=5, show_output=True)
```

```
testNetworkPing (scenario_name, machine_name, destination_machine_name, count=1)
```

### *Module contents*

## Indices and tables

- `genindex`
- `modindex`
- `search`

# Index

## A

`addSharedFolder()`  
(`Entities.VirtualMachine.VirtualMachine` method)  
`addVM()` (`Entities.Scenario.Scenario` method)

## C

`createMachineFolders()`  
(`Managers.FileManager.FileManager` method)  
`createScenario()` (in module `MainServer`)  
`createScenarioFolders()`  
(`Managers.FileManager.FileManager` method)  
`createVagrantFiles()`  
(`Managers.VagrantManager.VagrantManager` method)

## D

`DatabaseManager` (class in `Managers.DatabaseManager`)  
`deleteFile()` (in module `MainServer`)  
`deleteScenario()` (in module `MainServer`)  
    (`Managers.DatabaseManager.DatabaseManager` method)  
    (`Managers.ScenarioManager.ScenarioManager` method)  
`dictionary()` (`Entities.Entity.Entity` method)  
    (`Entities.ExploitInfo.ExploitInfo` method)  
    (`Entities.NetworkSettings.NetworkSettings` method)  
    (`Entities.Provision.Provision` method)  
    (`Entities.Response.Response` method)  
    (`Entities.Scenario.Scenario` method)  
    (`Entities.VirtualMachine.VirtualMachine` method)  
    (`Entities.VulnerabilityInfo.VulnerabilityInfo` method)

## E

`editScenario()` (in module `MainServer`)  
    (`Managers.DatabaseManager.DatabaseManager` method)  
    (`Managers.ScenarioManager.ScenarioManager` method)  
`enableGUI()` (`Entities.VirtualMachine.VirtualMachine` method)  
`Entities` (module)  
`Entities.Entity` (module)  
`Entities.ExploitInfo` (module)

`Entities.NetworkSettings` (module)  
`Entities.Provision` (module)  
`Entities.Response` (module)  
`Entities.Scenario` (module)  
`Entities.VagrantFile` (module)  
`Entities.VirtualMachine` (module)  
`Entities.VulnerabilityInfo` (module)  
`Entity` (class in `Entities.Entity`)  
`ExploitInfo` (class in `Entities.ExploitInfo`)

## F

`FileManager` (class in `Managers.FileManager`)

## G

`getAvailableBoxes()` (in module `MainServer`)  
    (`Managers.VagrantManager.VagrantManager` method)  
`getCurrentPath()` (`Managers.FileManager.FileManager` method)  
`getFileList()` (in module `MainServer`)  
`getJSONPath()` (`Managers.FileManager.FileManager` method)  
`getScenario()` (in module `MainServer`)  
    (`Managers.DatabaseManager.DatabaseManager` method)  
    (`Managers.ScenarioManager.ScenarioManager` method)  
`getScenarioNames()`  
(`Managers.DatabaseManager.DatabaseManager` method)  
`getScenarios()` (in module `MainServer`)  
    (`Managers.DatabaseManager.DatabaseManager` method)  
    (`Managers.ScenarioManager.ScenarioManager` method)  
`getScenariosPath()`  
(`Managers.FileManager.FileManager` method)

## H

`haltVM()` (`Managers.VagrantManager.VagrantManager` method)

## I

`insertScenario()`  
(`Managers.DatabaseManager.DatabaseManager` method)

## **M**

MainServer (module)  
Managers (module)  
Managers.DatabaseManager (module)  
Managers.FileManager (module)  
Managers.ScenarioManager (module)  
Managers.VagrantManager (module)

## **N**

NetworkSettings (class in Entities.NetworkSettings)  
newEmptyScenario()  
(Managers.ScenarioManager.ScenarioManager  
method)

## **O**

objectFromDictionary() (Entities.Entity.Entity method)  
(Entities.ExploitInfo.ExploitInfo method)  
(Entities.NetworkSettings.NetworkSettings  
method)  
(Entities.Provision.Provision method)  
(Entities.Response.Response method)  
(Entities.Scenario.Scenario method)  
(Entities.VirtualMachine.VirtualMachine method)  
(Entities.VulnerabilityInfo.VulnerabilityInfo method)

## **P**

Provision (class in Entities.Provision)

## **R**

Response (class in Entities.Response)  
restartVM()  
(Managers.VagrantManager.VagrantManager method)  
runVagrantUp() (in module MainServer)  
(Managers.VagrantManager.VagrantManager  
method)

## **S**

Scenario (class in Entities.Scenario)  
scenarioExists()  
(Managers.ScenarioManager.ScenarioManager  
method)  
ScenarioManager (class in  
Managers.ScenarioManager)  
sendCommand()  
(Managers.VagrantManager.VagrantManager method)  
setBody() (Entities.Response.Response method)

setCode() (Entities.Response.Response method)  
setExploitInfo() (Entities.Scenario.Scenario method)  
setName() (Entities.VirtualMachine.VirtualMachine  
method)  
setNetworkSettings()  
(Entities.VirtualMachine.VirtualMachine method)  
setOS() (Entities.VirtualMachine.VirtualMachine  
method)  
setProvision() (Entities.VirtualMachine.VirtualMachine  
method)  
setResponse() (Entities.Response.Response method)  
setShellCommand() (Entities.Provision.Provision  
method)  
setStatus() (Entities.Response.Response method)  
setTask\_id() (Entities.Response.Response method)  
setVulnerabilityInfo() (Entities.Scenario.Scenario  
method)

## **T**

testDB()  
(Managers.ScenarioManager.ScenarioManager  
method)  
testDB2()  
(Managers.ScenarioManager.ScenarioManager  
method)  
testNetworkPing()  
(Managers.VagrantManager.VagrantManager method)  
testPing() (in module MainServer)

## **U**

uploadFile() (in module MainServer)

## **V**

VagrantFile (class in Entities.VagrantFile)  
vagrantFilePerMachine()  
(Entities.VagrantFile.VagrantFile method)  
VagrantManager (class in Managers.VagrantManager)  
VirtualMachine (class in Entities.VirtualMachine)  
VulnerabilityInfo (class in Entities.VulnerabilityInfo)

# Python Module Index

## **e**

Entities

Entities.Entity

Entities.ExploitInfo

Entities.NetworkSettings

Entities.Provision

Entities.Response

Entities.Scenario

Entities.VagrantFile

Entities.VirtualMachine

Entities.VulnerabilityInfo

## **m**

MainServer

Managers

Managers.DatabaseManager

Managers.FileManager

Managers.ScenarioManager

Managers.VagrantManager