

Automated Malware and Exploit Data Collection Tool Documentation

version 1.0

Software Practicum Team 3

mayo 14, 2020

Contents

Welcome to Automated Malware and Exploit Data Collection Tool's documentation!	1
src	1
Entities package	1
Submodules	1
Entities.Entity module	1
Entities.Exploit module	1
Entities.NetworkSettings module	1
Entities.Program module	1
Entities.Provision module	2
Entities.Response module	2
Entities.Scenario module	2
Entities.VirtualMachine module	3
Entities.Vulnerability module	3
Module contents	4
MainServer module	4
Managers package	5
Submodules	5
Managers.ConfigManager module	5
Managers.DatabaseManager module	5
Managers.ExploitManager module	7
Managers.FileManager module	7
Managers.ScenarioManager module	8
Managers.VulnerabilityManager module	8
Module contents	9
Indices and tables	9
Index	11
Python Module Index	15

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

src

Entities package

Submodules

Entities.Entity module

```
class Entities.Entity.Entity
```

```
    Bases: abc.ABC
```

```
    abstract dictionary ()
```

```
        Converts an entity object to a dictionary.
```

```
    abstract objectFromDictionary (dict)
```

```
        Creates an object from a dictionary. :param dict: Dictionary containing the object information
```

Entities.Exploit module

```
class Entities.Exploit.Exploit (name="", type="", download_link="")
```

```
    Bases: Entities.Entity.Entity
```

```
    dictionary ()
```

```
        Generates a dictionary for the Exploit object :return: A dictionary with Exploit's data
```

```
    objectFromDictionary (dict)
```

```
        Creates an Exploit object from a dictionary. :param dict: A dictionary containing the Exploit's data :return: An Exploit object
```

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)
```

```
        Creates an NetworkSettings object from a dictionary. :param dict: A dictionary containing the NetworkSettings' data :return: A NetworkSettings object
```

Entities.Program module

```
class Entities.Program.Program (name="", location="")
```

```
    Bases: Entities.Entity.Entity
```

```
    dictionary ()
```

```
        Generates a dictionary for the Program object :return: A dictionary with Program's data
```

objectFromDictionary (dict)

Creates an Program object from a dictionary. :param dict: A dictionary containing the Program's data :return: A Program object

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's data

objectFromDictionary (dict)

Creates a Provision object from a dictionary. :param dict: A dictionary containing the Provision's data :return: A Provision object

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="", reason="", status="", task_id="", body="")
```

Bases: **Entities.Entity.Entity**

dictionary ()

Generates a dictionary for the Response object :return: A dictionary with Response's data

objectFromDictionary (dict)

Creates a Response object from a dictionary. :param dict: A dictionary containing the Response data :return: A Response object

setBody (body)

Sets the body of a request. :param: String containing the request's body

setReason (reason)

Sets success/fail reason of a request. :param reason: String containing the request's reason

setResponse (response)

Sets the response of a request. :param response: String containing the request's response

setStatus (status)

Set the status of a request. :param status: String containing the request's status

setTaskID (task_id)

Sets the task ID of a request. :param task_id: String containing the request's 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's data

objectFromDictionary (dict)

Creates a Scenario object from a dictionary. :param dict: A dictionary containing the Scenario's data :return: A Scenario object

setExploit (exploit)

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

setVulnerability (vulnerability)

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

Entities.VirtualMachine module

```
class Entities.VirtualMachine.VirtualMachine (name="", box='generic/alpine37', os="",
base_memory='1024', processors='2', is_attacker=False, shared_folders='./vmfiles', '/sharedfolder', uuid=")
```

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)

Creates a VirtualMachine object from a dictionary. :param dict: A dictionary containing the VirtualMachine's data :return: A VirtualMachine object

setBaseMemory (base_memory)

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 (i, provision)

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

Entities.Vulnerability module

```
class Entities.Vulnerability.Vulnerability (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)

Creates a Vulnerability object from a dictionary. :param dict: A dictionary containing the Vulnerability's data :return: A Vulnerability object

Module contents

MainServer module

MainServer.addBoxByName ()

MainServer.addBoxByOVAFile ()

MainServer.createExploit (exploit_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.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.createVulnerability (vulnerability_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.deleteExploit (exploit_name)
Edits a current scenario with a JSON file :param exploit_name: String with the scenario name :return: True if the scenario has been successfully edited, otherwise False

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.deleteVulnerability (vulnerability_name)
Edits a current scenario with a JSON file :param vulnerability_name: String with the scenario name :return: True if the scenario has been successfully edited, otherwise False

MainServer.editExploit ()
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.editVulnerability ()
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.getExploit (exploit_name)
Gets the scenario as a JSON file :param exploit_name: String with the scenario name :return: JSON file with the scenario info

MainServer.getExploits ()
Gets the available scenarios :return: A list of strings with the available scenarios

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.getSystemInfo ()

MainServer.getTaskStatus (task_id)

Requests the status of an ongoing task from the VagranServer :param task_id: Task ID given by Celery :return: a json response that denotes the status of the task

`MainServer.getVulnerabilities ()`

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

`MainServer.getVulnerability (vulnerability_name)`

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

`MainServer.removeBoxByName ()`

`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 ()`

`MainServer.vagrantCommand (scenario_name, machine_name, command)`

Managers package

Submodules

Managers.ConfigManager module

`class Managers.ConfigManager.ConfigManager`

Bases: `object`

`mongoURL ()`

Sets the machine's URL running the MongoDB database. :return: MongoDB's URL string

`redisURL ()`

Sets the machine's URL running Redis. :return: Redis' URL string

`uploadURL ()`

Sets the machine's URL running the upload cloud. :return: Upload's URL string

`vagrantURL ()`

Sets the machine's url running vagrant. :return: Vagrant's URL string

Managers.DatabaseManager module

`class Managers.DatabaseManager.DatabaseManager`

Bases: `object`

`addExploitsToDB ()`

Pre-populates the database with exploits. :return: None

`addScenariosToDB ()`

Pre-populates the database with scenarios. :return: None

`addVulnerabilitiesToDB ()`

Pre-populates the database with vulnerabilities. :return: None

`deleteExploit (exploit_name)`

Deletes a exploit from the database. :param exploit_name: Exploit's name string :return: Deleted document's id

deleteScenario (scenario_name)

Deletes a scenario from the database. :param scenario_name: Scenario's name string :return: Deleted document's id

deleteVulnerability (vulnerability_name)

Deletes a vulnerability from the database. :param vulnerability_name: Vulnerability's name string :return: Deleted document's id

editExploit (exploit_json)

Edits an exploit in the database. :param exploit_json: JSON file containing the exploit's data :return: Modified document's id

editScenario (scenario_json)

Edits a scenario in the database. :param scenario_json: JSON file containing scenario's data :return: Modified document's id

editVulnerability (vulnerability_json)

Edits a vulnerability in the database. :param vulnerability_json: JSON file containing the vulnerability's data :return: Modified document's id

getExploit (exploit_name)

Gets an exploit from the database. :param exploit_name: Exploit's name string :return: A list containing the exploit data

getExploitNames ()

Gets the exploits names from the database. :return: A list containing the exploits' names

getExploits ()

Gets the exploits stored in the database. :return: A list containing the stored exploits' data

getScenario (scenario_name)

Gets a specific scenario from the database. :param scenario_name: Scenario's name string :return: A list containing the scenario retrieved from the database

getScenarioNames ()

Gets the scenario's names. :return: A list containing the scenarios names

getScenarios ()

Gets the scenarios from the databases. :return: A list containing scenarios in the database

getVulnerabilities ()

Gets the vulnerabilities stored in the database. :return: A list containing the stored vulnerabilities' data

getVulnerability (vulnerability_name)

Gets a vulnerability from the database. :param vulnerability_name: Vulnerability's name string :return: A list containing the vulnerability's data

getVulnerabilityNames ()

Gets the vulnerabilities names from the database. :return: A list containing the vulnerability's names

insertExploit (exploit_json)

Inserts a scenario in the database. :param exploit_json: JSON file containing the exploit's data :return: Inserted document's id

insertScenario (scenario_json)

Inserts a scenario into the database. :param scenario_json: Scenario's JSON file to be inserted :return: Inserted document's id

insertVulnerability (vulnerability_json)

Inserts a vulnerability in the database. :param vulnerability_json: JSON file containing the vulnerability's data :return: Inserted document's id

Managers.ExploitManager module

```
class Managers.ExploitManager.ExploitManager
(db_manager=<Managers.DatabaseManager.DatabaseManager object>)
  Bases: object

  deleteOne (exploit_name)
    Deletes an exploit from the database. :param exploit_name: Exploit's name string :return: Response object
    containing the status of the request

  editOne (exploit_json)
    Edits a current scenario with a JSON file :param exploit_json: JSON file with the exploit's data :return: Response
    object containing the status of the request

  getAll ()
    Gets the available exploits :return: Response object containing the status of the request

  getOne (exploit_name)
    Gets the scenario as a JSON file :param exploit_name: Exploit's name string :return: Response object
    containing the status of the request

  newEmpty (exploit_name)
    Creates a new exploit which includes the folders and the exploit JSON file :param exploit_name: String with the
    exploit name :return: Response object containing the status of the request
```

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

  createSaltFiles (scenario_json)
    Creates the salt files per each machine in a scenario. :param scenario_json: JSON containing the scenario data
    :return: Response object containing request info

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

  createSharedFolders (scenario_json)
    Creates the shared folder within a scenario. :param scenario_json: JSON containing the scenario data :return:
    Response object containing request info

  createVagrantFiles (scenario_json)
    Creates a vagrant file per machine in a scenario. :param scenario_json: JSON containing the scenario data
    :return: Response object containing request info

  deleteScenariosFolder (scenario_name)
    Deletes not used scenario folders. :param scenario_name: Scenario name as a string :return: None

  getCurrentPath ()
    test Gets the project folder path :return: String with the project path
```

getExploitJSONPath (exploit_name)

Gets exploit JSON path. :param exploit_name: Exploit name to search :return: Exploit path

getExploitsPath ()

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

getScenarioJSONPath (scenario_name)

Gets scenario JSON path inside a machine. :param scenario_name: Scenario name string :return: JSON path folder

getScenariosPath ()

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

getVulnerabilityJSONPath (vulnerability_name)

Gets vulnerability JSON path. :param vulnerability_name: Vulnerability name to search :return: Vulnerability path

purgeMachines (scenario_name, safe_machines)

Deletes machines that no longer exist within the scenario. :param scenario_name: Scenario name as a string :param safe_machines: Collection containing machines that must remain within the scenario :return: None

Managers.ScenarioManager module

class Managers.ScenarioManager.ScenarioManager

(db_manager=<Managers.DatabaseManager.DatabaseManager object>)

Bases: **object**

deleteOne (scenario_name)

Deletes one scenario from the database. :param scenario_name: Scenario's name string :return: Response object containing the status of the request

editOne (scenario_json)

Edits a current scenario with a JSON file :param scenario_json: JSON file with the new scenario :return: Response object containing the status of the request

getAll ()

Gets the available scenarios :return: Response object containing the status of the request

getOne (scenario_name)

Gets the scenario as a JSON file :param scenario_name: String with the scenario name :return: Response object containing the status of the request

newEmpty (scenario_name)

Creates a new scenario which includes the folders and the scenario JSON file :param scenario_name: String with the scenario name :return: Response object containing the status of the request

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

Managers.VulnerabilityManager module

class Managers.VulnerabilityManager.VulnerabilityManager

(db_manager=<Managers.DatabaseManager.DatabaseManager object>)

Bases: **object**

deleteOne (vulnerability_name)

Deletes a vulnerability from the database. :param vulnerability_name: Vulnerability's name string :return: Response object containing the status of the request

editOne (vulnerability_json)

Edits a current scenario with a JSON file :param vulnerability_json: JSON file new vulnerability :return: Response object containing the status of the request

getAll ()

Gets the available exploits :return: Response object containing the status of the request

getOne (vulnerability_name)

Gets the scenario as a JSON file :param vulnerability_name: Vulnerability's name string :return: Response object containing the status of the request

newEmpty (vulnerability_name)

Creates a new vulnerability which includes the folders and the vulnerability JSON file :param vulnerability_name: String with the vulnerability name :return: Response object containing the status of the request

Module contents

Indices and tables

- **genindex**
- **modindex**
- **search**

Index

A

`addBoxByName()` (in module `MainServer`)
`addBoxByOVAFile()` (in module `MainServer`)
`addExploitsToDB()`
(`Managers.DatabaseManager.DatabaseManager` method)
`addScenariosToDB()`
(`Managers.DatabaseManager.DatabaseManager` method)
`addSharedFolder()`
(`Entities.VirtualMachine.VirtualMachine` method)
`addVM()` (`Entities.Scenario.Scenario` method)
`addVulnerabilitiesToDB()`
(`Managers.DatabaseManager.DatabaseManager` method)

C

`ConfigManager` (class in `Managers.ConfigManager`)
`createExploit()` (in module `MainServer`)
`createMachineFolders()`
(`Managers.FileManager.FileManager` method)
`createSaltFiles()` (`Managers.FileManager.FileManager` method)
`createSaltStackFolder()`
(`Managers.FileManager.FileManager` method)
`createScenario()` (in module `MainServer`)
`createScenarioFolders()`
(`Managers.FileManager.FileManager` method)
`createSharedFolders()`
(`Managers.FileManager.FileManager` method)
`createVagrantFiles()`
(`Managers.FileManager.FileManager` method)
`createVulnerability()` (in module `MainServer`)

D

`DatabaseManager` (class in `Managers.DatabaseManager`)
`deleteExploit()` (in module `MainServer`)
(`Managers.DatabaseManager.DatabaseManager` method)
`deleteFile()` (in module `MainServer`)
`deleteOne()`
(`Managers.ExploitManager.ExploitManager` method)
(`Managers.ScenarioManager.ScenarioManager` method)

(`Managers.VulnerabilityManager.VulnerabilityManager` method)

`deleteScenario()` (in module `MainServer`)

(`Managers.DatabaseManager.DatabaseManager` method)

`deleteScenariosFolder()`
(`Managers.FileManager.FileManager` method)

`deleteVulnerability()` (in module `MainServer`)

(`Managers.DatabaseManager.DatabaseManager` method)

`dictionary()` (`Entities.Entity.Entity` method)

(`Entities.Exploit.Exploit` method)

(`Entities.NetworkSettings.NetworkSettings` method)

(`Entities.Program.Program` method)

(`Entities.Provision.Provision` method)

(`Entities.Response.Response` method)

(`Entities.Scenario.Scenario` method)

(`Entities.VirtualMachine.VirtualMachine` method)

(`Entities.Vulnerability.Vulnerability` method)

E

`editExploit()` (in module `MainServer`)

(`Managers.DatabaseManager.DatabaseManager` method)

`editOne()` (`Managers.ExploitManager.ExploitManager` method)

(`Managers.ScenarioManager.ScenarioManager` method)

(`Managers.VulnerabilityManager.VulnerabilityManager` method)

`editScenario()` (in module `MainServer`)

(`Managers.DatabaseManager.DatabaseManager` method)

`editVulnerability()` (in module `MainServer`)

`editVulnerability()`
(`Managers.DatabaseManager.DatabaseManager` method)

`enableGUI()` (`Entities.VirtualMachine.VirtualMachine` method)

Entities

module

Entities.Entity

module

Entities.Exploit

module

Entities.NetworkSettings

module

Entities.Program

module

Entities.Provision

module

Entities.Response

module

Entities.Scenario

module

Entities.VirtualMachine

module

Entities.Vulnerability

module

Entity (class in Entities.Entity)

Exploit (class in Entities.Exploit)

ExploitManager (class in Managers.ExploitManager)

F

FileManager (class in Managers.FileManager)

G

getAll() (Managers.ExploitManager.ExploitManager method)

(Managers.ScenarioManager.ScenarioManager method)

(Managers.VulnerabilityManager.VulnerabilityManager method)

getAvailableBoxes() (in module MainServer)

getCurrentPath() (Managers.FileManager.FileManager method)

getExploit() (in module MainServer)

(Managers.DatabaseManager.DatabaseManager method)

getExploitJSONPath() (Managers.FileManager.FileManager method)

getExploitNames() (Managers.DatabaseManager.DatabaseManager method)

getExploits() (in module MainServer)

(Managers.DatabaseManager.DatabaseManager method)

getExploitsPath() (Managers.FileManager.FileManager method)

getFileList() (in module MainServer)

getOne() (Managers.ExploitManager.ExploitManager method)

(Managers.ScenarioManager.ScenarioManager method)

(Managers.VulnerabilityManager.VulnerabilityManager method)

getScenario() (in module MainServer)

(Managers.DatabaseManager.DatabaseManager method)

getScenarioJSONPath() (Managers.FileManager.FileManager method)

getScenarioNames() (Managers.DatabaseManager.DatabaseManager method)

getScenarios() (in module MainServer)

(Managers.DatabaseManager.DatabaseManager method)

getScenariosPath() (Managers.FileManager.FileManager method)

getSystemInfo() (in module MainServer)

getTaskStatus() (in module MainServer)

getVulnerabilities() (in module MainServer)

(Managers.DatabaseManager.DatabaseManager method)

getVulnerability() (in module MainServer)

(Managers.DatabaseManager.DatabaseManager method)

getVulnerabilityJSONPath() (Managers.FileManager.FileManager method)

getVulnerabilityNames() (Managers.DatabaseManager.DatabaseManager method)

I

insertExploit() (Managers.DatabaseManager.DatabaseManager method)

insertScenario() (Managers.DatabaseManager.DatabaseManager method)

insertVulnerability() (Managers.DatabaseManager.DatabaseManager method)

M

MainServer

module

Managers

module

Managers.ConfigManager

module

Managers.DatabaseManager

module

Managers.ExploitManager

module

Managers.FileManager

module

Managers.ScenarioManager

module

Managers.VulnerabilityManager

module

module

Entities

Entities.Entity

Entities.Exploit

Entities.NetworkSettings

Entities.Program

Entities.Provision

Entities.Response

Entities.Scenario

Entities.VirtualMachine

Entities.Vulnerability

MainServer

Managers

Managers.ConfigManager

Managers.DatabaseManager

Managers.ExploitManager

Managers.FileManager

Managers.ScenarioManager

Managers.VulnerabilityManager

mongoURL()
(Managers.ConfigManager.ConfigManager method)

N

NetworkSettings (class in Entities.NetworkSettings)

newEmpty()
(Managers.ExploitManager.ExploitManager method)

(Managers.ScenarioManager.ScenarioManager method)

(Managers.VulnerabilityManager.VulnerabilityManager method)

O

objectFromDictionary() (Entities.Entity.Entity method)

(Entities.Exploit.Exploit method)

(Entities.NetworkSettings.NetworkSettings method)

(Entities.Program.Program method)

(Entities.Provision.Provision method)

(Entities.Response.Response method)

(Entities.Scenario.Scenario method)

(Entities.VirtualMachine.VirtualMachine method)

(Entities.Vulnerability.Vulnerability method)

P

Program (class in Entities.Program)

Provision (class in Entities.Provision)

purgeMachines() (Managers.FileManager.FileManager method)

R

redisURL() (Managers.ConfigManager.ConfigManager method)

removeBoxByName() (in module MainServer)

Response (class in Entities.Response)

runVagrantUp() (in module MainServer)

S

Scenario (class in Entities.Scenario)

scenarioExists()
(Managers.ScenarioManager.ScenarioManager method)

ScenarioManager (class in Managers.ScenarioManager)

setBaseMemory()
(Entities.VirtualMachine.VirtualMachine method)

setBody() (Entities.Response.Response method)

setExploit() (Entities.Scenario.Scenario method)

setName() (Entities.VirtualMachine.VirtualMachine method)

setNetworkSettings()
(Entities.VirtualMachine.VirtualMachine method)

setOS() (Entities.VirtualMachine.VirtualMachine method)

setProvision() (Entities.VirtualMachine.VirtualMachine method)

setReason() (Entities.Response.Response method)

setResponse() (Entities.Response.Response method)

setShellCommand() (Entities.Provision.Provision method)

setStatus() (Entities.Response.Response method)

setTaskID() (Entities.Response.Response method)

setVulnerability() (Entities.Scenario.Scenario method)

T

testPing() (in module MainServer)

U

[uploadFile\(\)](#) (in module MainServer)

[uploadURL\(\)](#)
(Managers.ConfigManager.ConfigManager method)

V

[vagrantCommand\(\)](#) (in module MainServer)

[vagrantURL\(\)](#)
(Managers.ConfigManager.ConfigManager method)

[VirtualMachine](#) (class in Entities.VirtualMachine)

[Vulnerability](#) (class in Entities.Vulnerability)

[VulnerabilityManager](#) (class in Managers.VulnerabilityManager)

Python Module Index

e

Entities

Entities.Entity

Entities.Exploit

Entities.NetworkSettings

Entities.Program

Entities.Provision

Entities.Response

Entities.Scenario

Entities.VirtualMachine

Entities.Vulnerability

m

MainServer

Managers

Managers.ConfigManager

Managers.DatabaseManager

Managers.ExploitManager

Managers.FileManager

Managers.ScenarioManager

Managers.VulnerabilityManager