caldera

Apr 13, 2020

Usage Guide:

1 Getting started **3**
- 1.1 Autonomous red-team engagements . . . 3
- 1.2 Manual red-team engagements . . . 3
- 1.3 Autonomous incident-response . . . 4
- 1.4 Research on artificial intelligence . . . 4

2 Learning the terminology **5**
- 2.1 What is an agent? . . . 5
- 2.2 What is a group? . . . 5
- 2.3 What is an ability? . . . 5
- 2.4 What is an adversary? . . . 7
- 2.5 What is an operation? . . . 8
- 2.6 What is a fact? . . . 9
- 2.7 What is a source? . . . 9
- 2.8 What is a rule? . . . 9
  - 2.8.1 Subnets . . . 10
- 2.9 What is a planner? . . . 10
  - 2.9.1 The *Sequential* planner . . . 10
- 2.10 What is a plugin? . . . 10

3 Server configuration **13**
- 3.1 The existing default.yml . . . 13
- 3.2 Adding your own config file . . . 14

4 Plugin library **15**
- 4.1 Sandcat (54ndc47) . . . 15
  - 4.1.1 Deploy . . . 15
  - 4.1.2 Options . . . 15
- 4.2 Mock . . . 16
- 4.3 Terminal . . . 16
- 4.4 Stockpile . . . 16
- 4.5 Response . . . 17
- 4.6 Compass . . . 17
- 4.7 Caltack . . . 17
- 4.8 SSL . . . 17
- 4.9 Atomic . . . 18
- 4.10 GameBoard . . . 18

4.11 Human . . . 18
4.12 Training . . . 18
4.13 Access . . . 18

5 The REST API **19**
5.1 /api/rest . . . 19
5.1.1 DELETE . . . 19
5.1.2 PUT . . . 20
5.1.3 POST . . . 20
5.2 /file/upload . . . 20
5.2.1 Example . . . 20
5.3 /file/download . . . 20
5.3.1 Example . . . 21

6 How to Build Plugins **23**
6.1 Creating the structure . . . 23
6.2 The *enable* function . . . 23
6.3 Writing the code . . . 24
6.4 Making it visual . . . 24

7 How to Build Agents **27**
7.1 Understanding contacts . . . 27
7.2 Building an agent: HTTP contact . . . 27
7.2.1 Part #1 . . . 27
7.2.2 Part #2 . . . 28
7.2.3 Part #3 . . . 29
7.2.4 Part #4 . . . 29

8 How CALDERA makes decisions **31**

9 Initial Access **33**
9.1 Run an initial access technique . . . 33
9.2 Write an initial access ability . . . 33
9.2.1 Create a binary . . . 33
9.2.2 Create an ability . . . 34
9.2.3 Run the ability . . . 34

10 Install CALDERA offline **35**

11 Docker deployment **37**

12 Install script **39**
12.1 A Python virtual environment . . . 39
12.2 GoLang . . . 39
12.3 MinGW . . . 39

13 CALDERA 2.0 **41**
13.1 Adversary Mode . . . 41
13.2 Chain Mode . . . 41
13.3 What's the long-term plan? . . . 42
13.4 Why? . . . 42
13.5 Gotchas . . . 42

14 Common problems **43**
14.1 I'm getting an error starting the application! . . . 43
14.2 I start an agent but cannot see it from the server! . . . 43

14.3 I'm seeing issues in the browser - things don't seem right! . . . 43
14.4 I see a 404 when I try to download conf.yml! . . . 43
14.5 I ran an adversary and it didn't do everything! . . . 44

15 app **45**
15.1 app.api namespace . . . 45
15.1.1 Subpackages . . . 45
15.1.2 Submodules . . . 45
15.1.3 app.api.rest_api module . . . 45
15.2 app.contacts namespace . . . 46
15.2.1 Subpackages . . . 46
15.2.2 Submodules . . . 46
15.2.3 app.contacts.contact_gist module . . . 46
15.2.4 app.contacts.contact_html module . . . 47
15.2.5 app.contacts.contact_http module . . . 47
15.2.6 app.contacts.contact_tcp module . . . 47
15.2.7 app.contacts.contact_udp module . . . 47
15.2.8 app.contacts.contact_websocket module . . . 47
15.3 app.learning namespace . . . 48
15.3.1 Submodules . . . 48
15.3.2 app.learning.p_ip module . . . 48
15.3.3 app.learning.p_path module . . . 48
15.4 app.objects namespace . . . 48
15.4.1 Subpackages . . . 48
15.4.2 Submodules . . . 51
15.4.3 app.objects.c_ability module . . . 51
15.4.4 app.objects.c_adversary module . . . 51
15.4.5 app.objects.c_agent module . . . 51
15.4.6 app.objects.c_obfuscator module . . . 52
15.4.7 app.objects.c_operation module . . . 52
15.4.8 app.objects.c_planner module . . . 53
15.4.9 app.objects.c_plugin module . . . 54
15.4.10 app.objects.c_schedule module . . . 54
15.4.11 app.objects.c_source module . . . 54
15.5 app.service namespace . . . 54
15.5.1 Submodules . . . 54
15.5.2 app.service.app_svc module . . . 54
15.5.3 app.service.auth_svc module . . . 55
15.5.4 app.service.contact_svc module . . . 56
15.5.5 app.service.data_svc module . . . 56
15.5.6 app.service.file_svc module . . . 57
15.5.7 app.service.learning_svc module . . . 59
15.5.8 app.service.planning_svc module . . . 59
15.5.9 app.service.rest_svc module . . . 60
15.6 app.utility namespace . . . 61
15.6.1 Submodules . . . 61
15.6.2 app.utility.base_obfuscator module . . . 61
15.6.3 app.utility.base_object module . . . 61
15.6.4 app.utility.base_parser module . . . 61
15.6.5 app.utility.base_planning_svc module . . . 62
15.6.6 app.utility.base_service module . . . 63
15.6.7 app.utility.base_world module . . . 63
15.6.8 app.utility.file_decryptor module . . . 64
15.6.9 app.utility.payload_encoder module . . . 64

15.6.10 app.utility.rule_set module 64

16 Indices and tables **65**

Python Module Index **67**

Index **69**

If you're new to CALDERA, this is a good place to start.

CHAPTER 1

Getting started

Before you start using CALDERA, you should determine how you'd like to use it. Because it is a cyber security framework, there are several ways you can utilize it - from offensive (red) to defensive (blue).

Here are the most common use-cases and basic instructions on how to proceed.

1.1 Autonomous red-team engagements

This is the original CALDERA use-case. You can use the framework to build a specific threat (adversary) profile and launch it in a network to see where you may be susceptible. This is good for testing defenses and training blue teams on how to detect threats.

To use this:

1. Log in as a red user
2. Click into the Sandcat plugin and deploy an agent on any compromised host
3. Review or build threat profiles in the Profiles tab. Hint: getting experienced red-team operators to build these profiles allows the blue team to re-run them anytime they want in the future.
4. Launch an operation against your agents using any threat profile

All built-in threat profiles are safe to use out-of-the-box.

1.2 Manual red-team engagements

You can use CALDERA to perform manual red-team assessments by leveraging the terminal plugin. This is good for replacing or appending existing offensive toolsets in a manual assessment, as the framework can be extended with any custom tools you may have.

To use this:

1. Log in as a red user

2. Click into the terminal plugin and deploy the Manx agent on any compromised host
3. Use the created sessions inside the terminal emulator to perform manual commands

1.3 Autonomous incident-response

You can leverage CALDERA to perform automated incident response on a given host. This is helpful for identifying TTPs that other security tools may not see or block.

To use this:

1. Log in as a blue user
2. Click into the Sandcat plugin and deploy an agent on any host
3. Review or build defender profiles in the Profiles tab
4. Launch an operation against your agents using any defender profile

Defender profiles utilize fact sources to determine good vs. bad on a given host.

1.4 Research on artificial intelligence

You can ignore all red/blue and security aspects of CALDERA and instead use it to test artificial intelligence and other decision-making algorithms.

To use this:

1. Enable the mock plugin and restart the server. Log in as a red user.
2. In the Campaigns -> Agents tab, review the simulated agents that have been spun up
3. Run an operation using any adversary against your simulated agents. Note how the operation runs non-deterministically. You can now go into the sequential.py planning module and adjust the logic which makes decisions on what to do when to test out different theories.

CHAPTER 2

Learning the terminology

2.1 What is an agent?

An agent is a simple software program - requiring no installation - which connects to CALDERA in order to get instructions. It then executes the instructions and sends the results back.

CALDERA includes a plugin, Sandcat (54ndc47), which is our default agent.

2.2 What is a group?

A group is a collection of agents connected to CALDERA. A group allows you to start an operation against multiple computers at the same time instead of one at a time.

When an agent is started, it automatically joins a generic group, my_group, but you can add or remove custom groups once an agent is connected.

During an operation, if an agent laterally moves to another computer, it will automatically become associated with the same group as the agent which started it.

2.3 What is an ability?

An ability is a specific ATT&CK technique implementation (procedure). Abilities are stored in YML format and are loaded into CALDERA each time it starts.

All abilities are stored inside the Stockpile plugin, along with profiles which use them.

Here is a sample ability:

```
- id: 9a30740d-3aa8-4c23-8efa-d51215e8a5b9
  name: Scan WIFI networks
  description: View all potential WIFI networks on host
```

(continues on next page)

(continued from previous page)

```
  tactic: discovery
  technique:
    attack_id: T1016
    name: System Network Configuration Discovery
  platforms:
    darwin:
      sh:
        command: |
          ./wifi.sh scan
        payload: wifi.sh
    linux:
      sh:
        command: |
          ./wifi.sh scan
        payload: wifi.sh
    windows:
      psh,pwsh:
        command: |
          .\wifi.ps1 -Scan
        payload: wifi.ps1
```

Things to note:

- Each ability has a random UUID id
- Each ability requires a name, description, ATT&CK tactic and technique information
- Each ability requires a platforms list, which should contain at least 1 block for a supported operating system (platform). Currently, abilities can be created for darwin, linux or windows.

For each platform, there should be a list of executors. Currently Darwin and Linux platforms can use sh and Windows can use psh (PowerShell), cmd (command prompt) or pwsh (open-source PowerShell core).

Each platform block consists of a:

- command (required)
- payload (optional)
- cleanup (optional)
- parsers (optional)
- requirements (optional)

Command: A command can be 1-line or many and should contain the code you would like the ability to execute. The command can (optionally) contain variables, which are identified as #{variable}. In the example above, there is one variable used, #{files}. A variable means that you are letting CALDERA fill in the actual contents. CALDERA has 3 global variables:

- #{server} references the FQDN of the CALDERA server itself. Because every agent may know the location of CALDERA differently, using the #{server} variable allows you to let the system determine the correct location of the server.
- #{group} is the group a particular agent is a part of. This variable is mainly useful for lateral movement, where your command can start an agent within the context of the agent starting it.
- #{location} is the location of the agent on the client file system.
- #{paw} is the unique identifier - or paw print - of the agent

Global variables can be identified quickly because they will be single words.

You can use these global variables freely and they will be filled in before the ability is used. Alternatively, you can write in your own variables and supply CALDERA with facts to fill them in.

Payload: A comma-separated list of files which the ability requires in order to run. In the windows executor above, the payload is wifi.ps1. This means, before the ability is used, the agent will download wifi.ps1 from CALDERA. If the file already exists, it will not download it. You can store any type of file in the payload directories of any plugin.

> Did you know that you can assign functions to execute on the server when specific payloads are requested for download? An example of this is the sandcat.go file. Check the plugins/sandcat/hook.py file to see how special payloads can be handled.

Payloads can be stored as regular files or you can xor (encode) them so the anti-virus on the server-side does not pick them up. To do this, run the app/utility/payload_encoder.py against the file to create an encoded version of it. Then store and reference the encoded payload instead of the original.

> The payload_encoder.py file has a docstring which explains how to use the utility.

Cleanup: An instruction that will reverse the result of the command. This is intended to put the computer back into the state it was before the ability was used. For example, if your command creates a file, you can use the cleanup to remove the file. Cleanup commands run after an operation, in the reverse order they were created. Cleaning up an operation is also optional, which means you can start an operation and instruct it to skip all cleanup instructions.

Cleanup is not needed for abilities, like above, which download files through the payload block. Upon an operation completing, all payload files will be removed from the client (agent) computers.

Parsers: A list of parsing modules which can parse the output of the command into new facts. Interested in this topic? Check out *how CALDERA makes decisions* which goes into detail about parsers.

Abilities can also make use of two CALDERA REST API endpoints, file upload and download.

Requirements: Required relationships of facts that need to be established before this ability can be used.

2.4 What is an adversary?

An adversary is a collection of abilities.

The abilities inside an adversary can optionally be grouped into phases, which allows a user to choose which order they are executed. During an operation, each phase of the adversary is run in order. If there are multiple abilities in the same phase, CALDERA will determine which order to run them, based on the information it has gathered thus far in the operation. This decision making process is known as the planner. The main reason to group abilities into phases is if an ability from a latter phase depends on the fact output from a previous phase.

An adversary can contain abilities which can be used on any platform (operating system). As an operation runs an adversary, CALDERA will match each ability to each agent and only send the matching ones to the agent.

Adversaries can be built either through the GUI or by adding YML files into `data/adversaries/` which is in the Stockpile plugin.

An adversary YML file can include a `phases` section that lists the IDs of the abilities to execute in each phase. Here is an example of such an adversary:

```
id: 5d3e170e-f1b8-49f9-9ee1-c51605552a08
name: Collection
description: A collection adversary pack
phases:
  1:
    - 1f7ff232-ebf8-42bf-a3c4-657855794cfe #find company emails
    - d69e8660-62c9-431e-87eb-8cf6bd4e35cf #find ip addresses
```

(continues on next page)

(continued from previous page)

```
    - 90c2efaa-8205-480d-8bb6-61d90dbaf81b #find sensitive files
    - 6469befa-748a-4b9c-a96d-f191fde47d89 #create staging dir
```

An adversary can be included in another adversary as a pack of abilities. This can be used to organize ability phases into groups for reuse by multiple adversaries. To do so, put the ID of another adversary in a phase just like an ability. In this case, CALDERA will expand and complete all the phases of that adversary before moving to the next phase.

An adversary YML file can also contain a `packs` section that contains the IDs of other adversaries. The ability phases from these adversary packs will be merged together into any existing phases, whether from the `phases` section itself or from other adversaries in the `packs` section. Here is an example using packs without phases:

```
id: de07f52d-9928-4071-9142-cb1d3bd851e8
name: Hunter
description: Discover host details and steal sensitive files
packs:
  - 0f4c3c67-845e-49a0-927e-90ed33c044e0
  - 1a98b8e6-18ce-4617-8cc5-e65a1a9d490e
```

Adversary YML files must contain either `packs` or `phases`, or both.

2.5 What is an operation?

An operation is started when you point an adversary at a group and have it run all capable abilities.

An operation can be started with a number of optional configurations:

- **Group**: Which collection of agents would you like to run against
- **Adversary**: Which adversary profile would you like to run
- **Run immediately**: Run the operation immediately or start in a paused state
- **Planner**: You can select which logic library - or planner - you would like to use.
- **Fact source**: You can attach a source of facts to an operation. This means the operation will start with "pre-knowledge" of the facts, which it can use to fill in variables inside the abilities.
- **Autonomous**: Run autonomously or manually. Manual mode will ask the operator to approve or discard each command.
- **Phases**: Run the adversary normally, abiding by phases, or smash all phases into a single one.
- **Auto-close**: Automatically close the operation when there is nothing left to do. Alternatively, keep the operation forever.
- **Obfuscators**: Select an obfuscator to encode each command with, before they are sent to the agents.
- **Jitter**: Agents normally check in with CALDERA every 60 seconds. Once they realize they are part of an active operation, agents will start checking in according to the jitter time, which is by default 2/8. This fraction tells the agents that they should pause between 2 and 8 seconds (picked at random each time an agent checks in) before using the next ability.
- **Visibility**: How visible should the operation be to the defense. Defaults to 51 because each ability defaults to a visibility of 50. Abilities with a higher visibility than the operation visibility will be skipped.

2.6 What is a fact?

A fact is an identifiable piece of information about a given computer. Facts are directly related to variables, which can be used inside abilities.

Facts are composed of a:

- **trait**: a 3-part descriptor which identifies the type of fact. An example is host.user.name. A fact with this trait tells me that it is a user name. This format allows you to specify the major (host) minor (user) and specific (name) components of each fact.
- **value**: any arbitrary string. An appropriate value for a host.user.name may be "Administrator" or "John".
- **score**: an integer which associates a relative importance for the fact. Every fact, by default, gets a score of 1. If a host.user.password fact is important or has a high chance of success if used, you may assign it a score of 5. When an ability uses a fact to fill in a variable, it will use those with the highest scores first. If a fact has a score of 0, it will be blacklisted - meaning it cannot be used in the operation.

  If a property has a major component = host (e.g., host.user.name) that fact will only be used by the host that collected it.

As hinted above, when CALDERA runs abilities, it scans the command and cleanup instructions for variables. When it finds one, it then looks at the facts it has and sees if it can replace the variables with matching facts (based on the property). It will then create new variants of each command/cleanup instruction for each possible combination of facts it has collected. Each variant will be scored based on the cumulative score of all facts inside the command. The highest scored variants will be executed first.

2.7 What is a source?

A source is a collection of facts that you have grouped together. A fact source can be applied to an operation when you start it, which gives the operation facts to fill in variables with.

2.8 What is a rule?

A Rule is a way of restricting or placing boundaries on CALDERA. Rules are directly related to facts and should be included in a fact sheet.

Rules act similar to firewall rules and have three key components: fact, action, and match

1. **Fact** specifies the name of the fact that the rule will apply to.
2. **Action** (ALLOW,DENY) will allow or deny the fact from use if it matches the rule.
3. **Match** regex rule on a fact's value to determine if the rule applies.

During an operation, the planning service matches each link against the rule-set, discarding it if any of the fact assignments in the link match a rule specifying DENY and keeping it otherwise. In the case that multiple rules match the same fact assignment, the last one listed will be given priority.

Example

```
rules:
  - action: DENY
    fact: file.sensitive.extension
    match: .*
  - action: ALLOW
```

(continues on next page)

(continued from previous page)

```
    fact: file.sensitive.extension
    match: txt
```

In this example only the txt file extension will be used. Note that the ALLOW action for txt supersedes the DENY for all, as the ALLOW rule is listed later in the policy. If the ALLOW rule was listed first, and the DENY rule second, then all values (including txt) for file.sensitive.extension would be discarded.

2.8.1 Subnets

Rules can also match against subnets.

Subnet Example

```
- action: DENY
  fact: my.host.ip
  match: .*
- action: ALLOW
  fact: my.host.ip
  match: 10.245.112.0/24
```

In this example, the rules would permit CALDERA to only operate within the 10.245.112.1 to 10.245.112.254 range

2.9 What is a planner?

A planner is a module within CALDERA which contains logic for how a running operation should make decisions about which abilities to use and in what order.

Specifically, a planner's logic contains the decision making to execute a single phase of an operation. The core service - operation_svc.py - calls the planner for each phase of an adversary during an operation.

> Planners are single module Python files. Planners utilize the core system's planning_svc.py, which has planning logic useful for various types of planners.

2.9.1 The *Sequential* planner

CALDERA ships with a default planner, sequential. During each phase of the operation, the sequential planner loops through all agents (which are part of the operation's group) and sends each of them a list of all ability commands the planner thinks it can complete. This decision is based on the agent matching the operating system (execution platform) of the ability and the ability command having no unsatisfied variables. It then waits for each agent to complete their list of commands before moving on to the next phase.

2.10 What is a plugin?

CALDERA is built using a plugin architecture on top of the core system. Plugins are separate git repositories that plug new features into the core system. Each plugin resides in the plugins directory and is loaded into CALDERA by adding it to the default.yml file.

Each plugin contains a single hook.py file in its root directory. This file should contain an initialize function, which gets called automatically for each loaded plugin when CALDERA boots. The initialize function contains the plugin logic that is getting "plugged into" the core system. This function takes a single parameter:

1. **services**: a list of core services that live inside the core system.

A plugin can add nearly any new functionality/features to CALDERA by using the two objects above.

CHAPTER 3

Server configuration

Caldera's configuration file is located at `conf/default.yml`.

3.1 The existing default.yml

The YAML configuration file contains all the configuration CALDERA requires to boot up. An example configuration file is below:

```
port: 8888
plugins:
  - sandcat
  - stockpile
  - compass
  - terminal
  - response
users:
  red:
    admin: admin
    red: admin
  blue:
    blue: admin
api_key: ADMIN123
exfil_dir: /tmp
reports_dir: /tmp
crypt_salt: REPLACE_WITH_RANDOM_VALUE
app.contact.http: http://0.0.0.0:8888
app.contact.tcp: 0.0.0.0:7010
app.contact.udp: 0.0.0.0:7011
app.contact.websocket: 0.0.0.0:7012
```

A few key things to note:

- **Port**: the port you serve CALDERA on

- **Plugins**: the list of all loaded plugins. A plugin must be in this list to be available when CALDERA is running. Adding a plugin to this list will result in that plugin's hook.py file getting called when CALDERA boots up.
- **Users**: the username/password credentials of all accounts you want to access the CALDERA login page. Users can either be in the red or blue group.
- **API_KEY**: a password to use when accessing CALDERA programmatically.
- **Exfil_dir**: the directory to use when an ability exfiltrates files from the agent, sending them back to CALDERA. Any file(s) posted to the /file/upload endpoint will end up in this directory.
- **Reports_dir**: the directory to save all reports when the server shuts down
- **app.contact.http**: the http location you want HTTP agents (like Sandcat) to connect to.
- **app.contact.tcp**: the TCP socket you want reverse-shell agents (like Manx) to connect to.
- **app.contact.udp**: the UDP socket you want UDP agents (like Manx) to connect to
- **app.contact.websocket**: the websocket port agents can connect to

3.2 Adding your own config file

By default, CALDERA will use the `default.yml` file that is included with CALDERA, but this can be overridden by taking by creating your own `local.yml` file and saving it in the `conf/` directory. The name of the config file to use can also be specified with the `-E` flag when starting the server.

Caldera will choose the configuration file to use in the following order:

1. A config specified with the `-E` or `--environment` command-line options. For instance, if started with `python caldera.py -E foo`, CALDERA will load it's configuration from `conf/foo.yml`.
2. `conf/local.yml`: Caldera will prefer the local configuration file if no other options are specified.
3. `conf/default.yml`: If no config is specified with the `-E` option and it cannot find a `conf/local.yml` configuration file, CALDERA will use its default configuration options.

CHAPTER 4

Plugin library

Here you'll get a run-down of all open-source plugins, all of which can be found in the plugins/ directory as separate GIT repositories.

4.1 Sandcat (54ndc47)

The Sandcat plugin, otherwise known as 54ndc47, is the default agent that CALDERA ships with. 54ndc47 is written in GoLang for cross-platform compatibility.

54ndc47 agents require network connectivity to CALDERA at port 8888.

4.1.1 Deploy

To deploy 54ndc47, use one of the built-in delivery commands which allows you to run the agent on any operating system. Each of these commands downloads the compiled 54ndc47 executable from CALDERA and runs it immediately. Find the commands on the Sandcat plugin tab.

Once the agent is running, it should show log messages when it beacons into CALDERA.

> If you have GoLang installed on the CALDERA server, each time you run one of the delivery commands above, the agent will re-compile itself dynamically and it will change it's source code so it gets a different file hash (MD5) and a random name that blends into the operating system. This will help bypass file-based signature detections.

4.1.2 Options

When deploying a 54ndc47 agent, there are optional parameters you can use when you start the executable:

- **Server**: This is the location of CALDERA. The agent must have connectivity to this host/port.
- **Group**: This is the group name that you would like the agent to join when it starts. The group does not have to exist. A default group of my_group will be used if none is passed in.

- **v**: Use `-v` to see verbose output from sandcat. Otherwise, sandcat will run silently.

Customizing Default Options & Execution Without CLI Options

It's possible to customize the default values of these options when pulling Sandcat from the CALDERA server.This is useful if you want to hide the parameters from the process tree. You can do this by passing the values in as headers instead of as parameters.

For example, the following will download a linux executable that will use `http://10.0.0.2:8888` as the server address instead of `http://localhost:8888`.

```
curl -sk -X POST -H 'file:sandcat.go' -H 'platform:linux' -H 'server:http://10.0.0.
↪2:8888' http://localhost:8888/file/download > sandcat.sh
```

4.2 Mock

The Mock plugin adds a set of simulated agents to CALDERA and allows you to run complete operations without hooking any other computers up to your server.

These agents are created inside the `conf/agents.yml` file. They can be edited and you can create as many as you'd like. A sample agent looks like:

```
- paw: 1234
  username: darthvader
  host: deathstar
  group: simulation
  platform: windows
  location: C:\Users\Public
  enabled: True
  privilege: User
  c2: HTTP
  exe_name: sandcat.exe
  executors:
    - pwsh
    - psh
```

After you load the mock plugin and restart CALDERA, all simulated agents will appear as normal agents in the Chain plugin GUI and can be used in any operation.

4.3 Terminal

The terminal plugin adds reverse-shell capability to CALDERA, along with a TCP-based agent called Manx.

When this plugin is loaded, you'll get access to a new GUI page which allows you to drop reverse-shells on target hosts and interact manually with the hosts.

You can use the terminal emulator on the Terminal GUI page to interact with your sessions.

4.4 Stockpile

The stockpile plugin adds a few components to CALDERA:

- Abilities
- Adversaries
- Planner
- Facts

These components are all loaded through the data/* directory.

4.5 Response

The response plugin is an autonomous incident response plugin, which can fight back against adversaries on a compromised host.

4.6 Compass

Create visualizations to explore TTPs. Follow the steps below to create your own visualization:

1. Click 'Generate Layer'
2. Click '+' to open a new tab in the navigator
3. Select 'Open Existing Layer'
4. Select 'Upload from local' and upload the generated layer file

Compass leverages ATT&CK Navigator, for more information see: https://github.com/mitre-attack/attack-navigator

4.7 Caltack

The caltack plugin adds the public MITRE ATT&CK website to CALDERA. This is useful for deployments of CALDERA where an operator cannot access the Internet to reference the MITRE ATT&CK matrix.

After loading this plugin and restarting, the ATT&CK website is available from the CALDERA home page. Not all parts of the ATT&CK website will be available - but we aim to keep those pertaining to tactics and techniques accessible.

4.8 SSL

The SSL plugin adds HTTPS to CALDERA.

> This plugin only works if CALDERA is running on a Linux or MacOS machine. It requires HaProxy (>= 1.8) to be installed prior to using it.

When this plugin has been loaded, CALDERA will start the HAProxy service on the machine and then serve CALDERA at hxxps://[YOUR_IP]:8443, instead of the normal hxxp://[YOUR_IP]:8888.

CALDERA will **only** be available at https://[YOUR_IP]:8443 when using this plugin. All deployed agents should use the correct address to connect to CALDERA.

4.9 Atomic

The Atomic plugin imports all Red Canary Atomic tests from their open-source GitHub repository.

4.10 GameBoard

The GameBoard plugin allows you to monitor both red-and-blue team operations. The game tracks points for both sides and determines which one is "winning".

4.11 Human

The Human plugin allows you to build "Humans" that will perform user actions on a target system as a means to obfuscate red actions by Caldera. Each human is built for a specific operating system and leverages the Chrome browser along with other native OS applications to perform a variety of tasks. Additionally, these humans can have various aspects of their behavior "tuned" to add randomization to the behaviors on the target system.

4.12 Training

This plugin allows a user to gain a "User Certificate" which proves their ability to use CALDERA. This is the first of several certificates planned in the future. The plugin takes you through a capture-the-flag style certification course, covering all parts CALDERA.

4.13 Access

This plugin allows you to task any agent with any ability from the database. It also allows you to conduct initial access attacks.

CHAPTER 5

The REST API

All REST API functionality can be viewed in the rest_api.py module in the source code.

5.1 /api/rest

You can interact with all parts of CALDERA through the core REST API endpoint /api/rest. If you send requests to "localhost" - you are not required to pass a key header. If you send requests to 127.0.0.1 or any other IP addresses, the key header is required. You can set the API key in the conf/default.yml file. Some examples below will use the header, others will not, for example.

> Any request to this endpoint must include an "index" as part of the request, which routes it to the appropriate object type.

Here are the available REST API functions:

5.1.1 DELETE

Agents

Delete any agent.

```
curl -X DELETE http://localhost:8888/api/rest -d '{"index":"agents","paw":"$agent_paw
↪"}'
```

Operations

Delete any operation.

```
curl -X DELETE http://localhost:8888/api/rest -d '{"index":"operations","id":"
↪$operation_id"}'
```

5.1.2 PUT

Create a new operation. All that is required is the operation name, similar to creating a new operation in the browser.

```
curl -X PUT -H "KEY:$KEY" http://127.0.0.1:8888/api/rest -d '{"index":"operations",
↪"name":"testoperation1"}'
```

Optionally, you can include:

1. group (defaults to empty string)
2. adversary_id (defaults to empty string)
3. planner (defaults to *sequential*)
4. source (defaults to *basic*')
5. jitter (defaults to *2/8*)
6. obfuscator (defaults to *plain-text*)
7. visibility (defaults to *50*)
8. autonomous (defaults to *1*)
9. phases_enabled (defaults to *1*)
10. auto_close (defaults to *0*)

To learn more about these options, read the "What is an operation?" documentation section.

5.1.3 POST

5.2 /file/upload

Files can be uploaded to CALDERA by POST'ing a file to the /file/upload endpoint. Uploaded files will be put in the exfil_dir location specified in the default.yml file.

5.2.1 Example

```
curl -F 'data=@path/to/file' http://localhost:8888/file/upload
```

5.3 /file/download

Files can be dowloaded from CALDERA through the /file/download endpoint. This endpoint requires an HTTP header called "file" with the file name as the value. When a file is requested, CALDERA will look inside each of the payload directories listed in the local.yml file until it finds a file matching the name.

Files can also be downloaded indirectly through the payload block of an ability.

> Additionally, the 54ndc47 plugin delivery commands utilize the file download endpoint to drop the agent on a host

5.3.1 Example

```
curl -X POST -H "file:wifi.sh" http://localhost:8888/file/download > wifi.sh
```

CHAPTER 6

How to Build Plugins

Building your own plugin allows you to add custom functionality to CALDERA.

A plugin can be nearly anything, from a RAT/agent (like 54ndc47) to a new GUI or a collection of abilities that you want to keep in "closed-source".

Plugins are stored in the plugins directory. If a plugin is also listed in the local.yml file, it will be loaded into CALDERA each time the server starts. A plugin is loaded through its hook.py file, which is "hooked" into the core system via the server.py (main) module.

This walkthrough assumes you're pulling from the master branch.

6.1 Creating the structure

Start by creating a new directory called "abilities" in CALDERA's plugins directory. In this directory, create a hook.py file and ensure it looks like this:

```
name = 'Abilities'
description = 'A sample plugin for demonstration purposes'
address = None


async def enable(services):
    pass
```

The name should always be a single word, the description a phrase, and the address should be None, unless your plugin exposes new GUI pages. Our example plugin will be called "abilities".

6.2 The *enable* function

The enable function is what gets hooked into CALDERA at boot time. This function accepts one parameter:

1. **services**: a list of core services that CALDERA creates at boot time, which allow you to interact with the core system in a safe manner.

Core services can be found in the app/services directory.

6.3 Writing the code

Now it's time to fill in your own enable function. Let's start by appending a new REST API endpoint to the server. When this endpoint is hit, we will direct the request to a new class (AbilityFetcher) and function (get_abilities). The full hook.py file now looks like:

```
from aiohttp import web

name = 'Abilities'
description = 'A sample plugin for demonstration purposes'
address = None


async def enable(services):
    app = services.get('app_svc').application
    fetcher = AbilityFetcher(services)
    app.router.add_route('*', '/get/abilities', fetcher.get_abilities)


class AbilityFetcher:

    def __init__(self, services):
        self.services = services

    async def get_abilities(self, request):
        abilities = await self.services.get('data_svc').locate('abilities')
        return web.json_response(dict(abilities=[a.display for a in abilities]))
```

Now that our initialize function is filled in, let's add the plugin to the default.yml file and restart CALDERA. Once running, in a browser or via cURL, navigate to 127.0.0.1:8888/get/abilities. If all worked, you should get a JSON response back, with all the abilities within CALDERA.

6.4 Making it visual

Now we have a usable plugin, but we want to make it more visually appealing.

Start by creating a "templates" directory inside your plugin directory (abilities). Inside the templates directory, create a new file called abilities.html. Ensure the content looks like:

```
<div id="abilities-new-section" class="section-profile">
    <div class="row">
        <div class="topleft duk-icon"><img onclick="removeSection('abilities-new-
↪section')" src="/gui/img/x.png"></div>
        <div class="column section-border" style="flex:25%;text-align:left;
↪padding:15px;">
            <h1 style="font-size:70px;margin-top:-20px;">Abilities</h1>
        </div>
        <div class="column" style="flex:75%;padding:15px;text-align: left">
            <div>
```

(continues on next page)

(continued from previous page)

```
                {% for a in abilities %}
                    <pre style="color:white">{{ a }}</pre>
                    <hr>
                {% endfor %}
            </div>
        </div>
    </div>
</div>
```

Then, back in your hook.py file, let's fill in the address variable and ensure we return the new abilities.html page when a user requests 127.0.0.1/get/abilities. Here is the full hook.py:

```
from aiohttp_jinja2 import template, web

name = 'Abilities'
description = 'A sample plugin for demonstration purposes'
address = '/plugin/abilities/gui'

async def enable(services):
    app = services.get('app_svc').application
    fetcher = AbilityFetcher(services)
    app.router.add_route('*', '/plugin/abilities/gui', fetcher.splash)
    app.router.add_route('GET', '/get/abilities', fetcher.get_abilities)


class AbilityFetcher:
    def __init__(self, services):
        self.services = services
        self.auth_svc = services.get('auth_svc')

    async def get_abilities(self, request):
        abilities = await self.services.get('data_svc').locate('abilities')
        return web.json_response(dict(abilities=[a.display for a in abilities]))

    @template('abilities.html')
    async def splash(self, request):
        await self.auth_svc.check_permissions(request)
        abilities = await self.services.get('data_svc').locate('abilities')
        return(dict(abilities=[a.display for a in abilities]))
```

Restart CALDERA and navigate to the home page. Be sure to run `server.py` with the `--fresh` flag to flush the previous object store database.

You should see a new "abilities" tab at the top, clicking on this should navigate you to the new abilities.html page you created.

CHAPTER 7

How to Build Agents

Building your own agent is a way to create a unique - or undetectable - footprint on compromised machines. Our default agent, 54ndc47, is a representation of what an agent can do. This agent is written in GoLang and offers an extensible collection of command-and-control (C2) protocols, such as communicating over HTTP or GitHub Gist.

You can extend 54ndc47 by adding your own C2 protocols in place or you can follow this guide to create your own agent from scratch.

7.1 Understanding contacts

Agents are processes which are deployed on compromised hosts and connect with the C2 server periodically for instructions. An agent connects to the server through a *contact*, which is a specific connection point on the server.

Each contact is defined in an independent Python module and is registered with the contact_svc when the server starts.

There are currently several built-in contacts available: http, tcp, udp and websocket.

7.2 Building an agent: HTTP contact

Start by getting a feel for the HTTP endpoint, which are located in the contacts/contact_http.py module.

```
POST  /beacon
```

7.2.1 Part #1

Start by writing a POST request to the /beacon endpoint.

In your agent code, create a flat JSON dictionary of key/value pairs and ensure the following properties are included as keys. Add values which correlate to the host your agent will be running on. Note - all of these properties are optional - but you should aim to cover as many as you can.

If you don't include a platform and executors then the server will never provide instructions to the agent, as it won't know which ones are valid to send.

- **server**: The location (IP or FQDN) of the C2 server
- **platform**: The operating system
- **host**: The hostname of the machine
- **username**: The username running the agent
- **architecture**: The architecture of the host
- **executors**: A list of executors allowed on the host
- **privilege**: The privilege level of the agent process, either User or Elevated
- **pid**: The process identifier of the agent
- **location**: The location of the agent on disk
- **exe_name**: The name of the agent binary file

At this point, you are ready to make a POST request with the profile to the /beacon endpoint. You should get back:

1. The recommended number of seconds to sleep before sending the next beacon
2. The recommended number of seconds (watchdog) to wait before killing the agent, once the server is unreachable (0 means infinite)
3. A list of instructions - base64 encoded.

```
profile=$(echo '{"server":"http://127.0.0.1:8888","platform":"darwin","executors":["sh
↪"]}' | base64)
curl -s -X POST -d $profile localhost:8888/beacon | base64 --decode
...{"paw": "dcoify", sleep": 59, "watchdog": 0, "instructions": "[...]"}
```

The paw property returned back from the server represents a unique identifier for your new agent. Each time you call the /beacon endpoint without this paw, a new agent will be created on the server - so you should ensure that future beacons include it.

You can now navigate to the CALDERA UI, click into the agents tab and view your new agent.

7.2.2 Part #2

Now it's time to execute the instructions.

Looking at the previous response, you can see each instruction contains:

- **id**: The link ID associated to the ability
- **sleep**: A recommended pause to take after running this instruction
- **command**: A base64 encoded command to run
- **executor**: The executor to run the command under
- **timeout**: How long to let the command run before timing it out
- **payload**: A payload file name which must be downloaded before running the command, if applicable

Now, you'll want to revise your agent to loop through all the instructions, executing each command and POSTing the response back to the /beacon endpoint. You should pause after running each instruction, using the sleep time provided inside the instruction.

```
data=$(echo '{"result":{"id":$id, "output":$output, "status": $status, "pid":$pid}}' |
↪ base64)
curl -s -X POST -d $data localhost:8888/beacon
sleep $instruction_sleep
```

The POST details inside the result are as follows:

- **id**: the ID of the instruction you received
- **output**: the base64 encoded output from running the instruction
- **status**: the status code from running the instruction. If unsure, put 0.
- **pid**: the process identifier the instruction ran under. If unsure, put 0.

Once all instructions are run, the agent should sleep for the specified time in the beacon before calling the /beacon endpoint again. This process should repeat forever.

7.2.3 Part #3

Inside each instruction, there is an optional *payload* property that contains a filename of a file to download before running the instruction. To implement this, add a file download capability to your agent, directing it to the /file/download endpoint to retrieve the file:

```
payload='some_file_name.txt"
curl -X POST -H "file:$payload" http://localhost:8888/file/download > some_file_name.
↪txt
```

7.2.4 Part #4

You should implement the watchdog configuration. This property, passed to the agent in every beacon, contains the number of seconds to allow a dead beacon before killing the agent.

CHAPTER 8

How CALDERA makes decisions

CALDERA makes decisions using parsers, which are optional blocks inside an ability.

Let's look at an example snippet of an ability that uses a parser:

```
    darwin:
      sh:
        command: |
          find /Users -name '*.#{file.sensitive.extension}' -type f -not -path '*/\.*
↪' 2>/dev/null
        parsers:
          plugins.stockpile.app.parsers.basic:
            - source: host.file.sensitive
              edge: has_extension
              target: file.sensitive.extension
```

A parser is identified by the module which contains the code to parse the command's output. The parser can contain:

Source (required): A fact to create for any matches from the parser

Edge (optional): A relationship between the source and target. This should be a string.

Target (optional): A fact to create which the source connects too.

In the above example, the output of the command will be sent through the plugins.stockpile.app.parsers.basic module, which will create a relationship for every found file.

CHAPTER 9

Initial Access

CALDERA allows for easy initial access attacks, by leveraging the Access plugin. This guide will walk you through how to fire off an initial access attack, as well as how to build your own.

9.1 Run an initial access technique

Start by deploying an agent locally. This agent will be your "assistant". It will execute any attack you feed it. You could alternatively deploy the agent remotely, which will help mask where your initial access attacks are originating.

From the Access plugin, select your agent and either the initial access tactic or any pre-ATT&CK tactic. This will filter the abilities. Select any ability within your chosen tactic.

Once selected, a pop-up box will show you details about the ability. You'll need to fill in values for any properties your selected ability requires. Click OK when done.

Finally, click to run the ability against your selected agent. The ability will be in one of 3 states: IN-PROGRESS, SUCCESS or FAILED. If it is in either of the latter two states, you can view the logs from the executed ability by clicking on the star.

9.2 Write an initial access ability

You can easily add new initial access or pre-ATT&CK abilities yourself.

9.2.1 Create a binary

You can use an existing binary or write your own - in any language - to act as your payload. The binary itself should contain the code to execute your attack. It can be as simple or complex as you'd like. It should accept parameters for any dynamic behaviors. At minimum, you should require a parameter for "target", which would be your intended IP address, FQDN or other target that your attack will run against.

As an example, look at the scanner.sh binary used for conducting a simple NMAP scan:

```
#!/bin/bash

echo '[+] Starting basic NMAP scan'
nmap -Pn $1
echo '[+] Complete with module'
```

This binary simply echos a few log statements and runs an NMAP scan against the first parameter (i.e., the target) passed to it.

9.2.2 Create an ability

With your binary at hand, you can now create a new ability YML file inside the Access plugin (plugins/access/data/abilities/*). Select the correct tactic directory (or create one if one does not exist). Here is what the YML file looks like for the scanner.sh binary:

```
---
- id: 567eaaba-94cc-4a27-83f8-768e5638f4e1
  name: NMAP scan
  description: Scan an external host for open ports and services
  tactic: technical-information-gathering
  technique:
    name: Conduct active scanning
    attack_id: T1254
  platforms:
    darwin,linux:
      sh:
        command: |
          ./scanner.sh #{target.ip}
        timeout: 300
        payloads:
          - scanner.sh
```

This is the same format that is used for other CALDERA abilities, so refer to the "Learning the terminology" doc page for a run-through of all the fields.

9.2.3 Run the ability

With your ability YML file loaded, restart CALDERA and head to the Access plugin to run it.

CHAPTER 10

Install CALDERA offline

To install CALDERA on a server without internet access, `pip` can be used to download the CALDERA dependencies from a machine with internet access. Once the dependencies are downloaded, they can be copied to the offline machine and installed.

The internet machine's platform and python version should match offline server. For example, if the the offline target machine runs Python 3.6 on CentOS 7, then Python3.6 and CentOS 7 should be used to perform the packaging to minimize problems.

```
git clone --recursive https://github.com/mitre/caldera.git
mkdir caldera/python_deps
pip download -r caldera/requirements.txt --dest caldera/python_deps
```

The `caldera` directory can now be copied to the offline server via whatever means are convenient (`scp` if there's connectivity, sneakernet, etc)

Once the `caldera` directory has been copied to the offline machine the dependencies can be installed with pip.

```
pip install -r caldera/requirements.txt --no-index --find-links caldera/python_deps
```

CALDERA can then be started as usual:

```
cd caldera
python server.py
```

CHAPTER 11

Docker deployment

If you wish to run CALDERA from a Docker container, execute the commands below.

1. Build a container from the latest changes.

```
docker build . -t caldera:server
```

1. Run the docker CALDERA server

```
docker run -p 7010:7010 -p 7011:7011 -p 7012:7012 -p 8888:8888 caldera:server
```

CHAPTER 12

Install script

If you would like to install CALDERA quickly using our recommended approach, you should use the install.sh script, found at the root of the project. This script can be run on MacOS, Ubuntu and CentOS computers:

```
./install.sh --darwin
./install.sh --centos
./install.sh --ubuntu
./install.sh --kali
```

This installer will additionally install:

12.1 A Python virtual environment

A new virtualenv for Python will be created at the root of the project, called calderaenv. All PIP requirements will be installed here instead of the host machine directly.

12.2 GoLang

This is used to dynamically compile the 54ndc47 agent every time it is requested, giving it a new file hash each time. If GoLang is not installed on the server, the agent will be downloaded from the sandcat/payloads directory and will not be dynamically compiled.

> If GoLang is installed, you can dynamically compile any Go payload per request by simply utilizing the file_svc:compile_go function. You'll see examples of this in the sandcat and terminal plugins.

12.3 MinGW

MinGW enables gcc compiling support for windows platforms. CALDERA uses MinGW to build C-Shared library (DLLs) versions of Sandcat.

CHAPTER 13

CALDERA 2.0

In April 2019, the CALDERA team pushed out several large changes to the core CALDERA code-base, effectively creating a "new" version of CALDERA: CALDERA 2.0. This new version of CALDERA featured a much more modular plugin-based architecture, many bug-fixes, a new GUI, and most importantly, the introduction of two operating modes: **adversary mode** and **chain mode**.

13.1 Adversary Mode

Adversary mode is the classic CALDERA capability. Functionally, the mode is the same as it was when first released in 2017 – it still runs fully automated, end-to-end operations using a dynamic planner – although now it has some internal optimizations, bug fixes, and a different GUI. Setup and requirements for this mode are also largely the same as when first released: you must install the CAgent software on each endpoint you want to test (Windows only), pointing the agent back to the CALDERA server to run the operation. Installing the agent is now much simpler, and can be done via a PowerShell script that's displayed on the adversary mode page. From an architecture perspective, the adversary mode functionality is now entirely encapsulated in the "adversary" plugin; without loading this plugin, the functionality will be absent.

13.2 Chain Mode

Chain mode is the new operating mode for CALDERA, and was first introduced in the "2.0" release in mid 2019. This mode was designed to allow users to orchestrate/string together atomic unit tests into larger attack sequences; unlike adversary mode, chain mode was originally not designed to be dynamic, and each operation was to be run explicitly sans any variables in commands. Chain mode's relatively simple use case enabled us to design it with a much smaller footprint, requiring only simple agents to execute commands as dictated by the CALDERA server; unlike adversary mode, chain mode leverages a single agent (not an agent + a RAT), and only needs a single agent to be connected to the CALDERA server to test a network (as opposed to each endpoint needing to have CAgent installed). Generally speaking, chain mode has significantly less overhead than adversary mode, albeit at the cost of some of adversary mode's dynamism.

13.3 What's the long-term plan?

Long term, we hope to subsume adversary mode's capabilities into chain mode by adding dynamism to chain mode operations, encoding input and output for each chain mode action in a way that's similar to (though more intuitive than) the way actions are encoded in adversary mode.

13.4 Why?

After releasing the first version of chain mode, we realized that this new functionality was significantly easier to stand up than our initial adversary mode release; we've found that many people struggling to run adversary mode operations are typically struggling with that mode's dependencies/encoding. Moreover, chain mode's light overhead makes it easier to extend with new actions, allowing us to more readily encode more of the ATT&CK matrix than we could with adversary mode. We believe that shifting our operations to something lighter-weight will allow more people to use CALDERA for more use cases.

13.5 Gotchas

Our CALDERA 2.0 push came without much fanfare – or documentation! We've discovered that there are some minor pain points when first using this new version:

- Adversary mode is disabled by default. To use adversary mode, download the adversary mode plugin and make the appropriate changes to the main CALDERA local.conf file.
- Existing documentation on CALDERA is largely out to date. In particular, our page on readthedocs needs to be updated. Much of that information still pertains to adversary mode, although stuff that talks more broadly about CALDERA is somewhat dated.
- CALDERA's GUI is now significantly different; don't worry if it doesn't look the way it does in other public material!
- Adversary mode still only supports execution on Windows machines. Chain mode by contrast has support for Windows, Linux, or Mac.

CHAPTER 14

Common problems

14.1 I'm getting an error starting the application!

1. You likely have a misalignment between the core code and the plugin repositories. This is typically caused by doing a "git pull" on one of the repositories and not the others. If you see this error, you should re-clone the latest stable version of CALDERA (recursively, of course).
2. Another common reason for this is running CALDERA from < Python 3.6.1.

14.2 I start an agent but cannot see it from the server!

1. Check that the firewall is open, allowing network connections, between the remote computer running the agent and the server itself.
2. Ensure you are serving CALDERA on all interfaces (0.0.0.0).
3. If you are running the agent on Windows, our default agent assumes it is a 64-bit system. If you are running 32-bit, you'll need to recompile the Windows agent to use it.

14.3 I'm seeing issues in the browser - things don't seem right!

1. Are you using Chrome or Safari? These are the only supported/tested browsers. All other ones are use-at-your-own-risk.

14.4 I see a 404 when I try to download conf.yml!

1. The conf.yml file is only relevant to pre-CALDERA 2.0. You should go to the README page and follow the instructions to run one of the missions.

14.5 I ran an adversary and it didn't do everything!

1. Check each ability on the adversary profile. It should show an icon for which operating system it runs on. Match this up with the operating systems of your agents. These are the only abilities an operation will attempt to run.
2. Look at each ability command. If there is a variable inside - shown by #{} syntax - the ability will need to be "unlocked" by another ability, in a prior phase, before it can run.

The following section contains information intended to help developers understand the inner workings of the CALDERA adversary emulation tool, CALDERA plugins, or new tools that interface with the CALDERA server.

CHAPTER 15

app

15.1 app.api namespace

15.1.1 Subpackages

app.api.packs namespace

Submodules

app.api.packs.advanced module

class app.api.packs.advanced.**AdvancedPack**(*services*)
Bases: *app.utility.base_world.BaseWorld*

enable()

app.api.packs.campaign module

class app.api.packs.campaign.**CampaignPack**(*services*)
Bases: *app.utility.base_world.BaseWorld*

enable()

15.1.2 Submodules

15.1.3 app.api.rest_api module

class app.api.rest_api.**RestApi**(*services*)
Bases: *app.utility.base_world.BaseWorld*

download_file(*request*)

enable()

landing(*request*)

login(*request*)

logout(*request*)

rest_core(***params*)

upload_file(*request*)

validate_login(*request*)

15.2 app.contacts namespace

15.2.1 Subpackages

app.contacts.handles namespace

Submodules

app.contacts.handles.h_beacon module

class app.contacts.handles.h_beacon.**Handle**(*tag*)
Bases: object

static run(*message*, *services*, *caller*)

15.2.2 Submodules

15.2.3 app.contacts.contact_gist module

class app.contacts.contact_gist.**Gist**(*services*)
Bases: *app.utility.base_world.BaseWorld*

get_beacons()
Retrieve all GIST beacons for a particular api key :return: the beacons

get_results()
Retrieve all GIST posted results for a this C2's api key :return:

gist_operation_loop()

handle_beacons(*beacons*)
Handles various beacons types (beacon and results)

retrieve_config()

start()

valid_config()

app.contacts.contact_gist.**api_access**(*func*)

15.2.4 app.contacts.contact_html module

class app.contacts.contact_html.**Html**(*services*)
Bases: *app.utility.base_world.BaseWorld*

start()

15.2.5 app.contacts.contact_http module

class app.contacts.contact_http.**Http**(*services*)
Bases: *app.utility.base_world.BaseWorld*

start()

15.2.6 app.contacts.contact_tcp module

class app.contacts.contact_tcp.**Tcp**(*services*)
Bases: *app.utility.base_world.BaseWorld*

operation_loop()

start()

class app.contacts.contact_tcp.**TcpSessionHandler**(*services*, *log*)
Bases: *app.utility.base_world.BaseWorld*

accept(*reader*, *writer*)

refresh()

send(*session_id*, *cmd*)

15.2.7 app.contacts.contact_udp module

class app.contacts.contact_udp.**Handler**(*services*)
Bases: asyncio.protocols.DatagramProtocol

datagram_received(*data*, *addr*)
Called when some datagram is received.

class app.contacts.contact_udp.**Udp**(*services*)
Bases: *app.utility.base_world.BaseWorld*

start()

15.2.8 app.contacts.contact_websocket module

class app.contacts.contact_websocket.**Handler**(*services*)
Bases: object

handle(*socket*, *path*)

class app.contacts.contact_websocket.**WebSocket**(*services*)
Bases: *app.utility.base_world.BaseWorld*

start()

15.3 app.learning namespace

15.3.1 Submodules

15.3.2 app.learning.p_ip module

class app.learning.p_ip.**Parser**
Bases: object

parse(*blob*)

15.3.3 app.learning.p_path module

class app.learning.p_path.**Parser**
Bases: object

parse(*blob*)

15.4 app.objects namespace

15.4.1 Subpackages

app.objects.secondclass namespace

Submodules

app.objects.secondclass.c_fact module

class app.objects.secondclass.c_fact.**Fact**(*trait*, *value*, *score=1*, *collected_by=None*, *technique_id=None*)
Bases: *app.utility.base_object.BaseObject*

display

escaped(*executor*)

unique

app.objects.secondclass.c_instruction module

class app.objects.secondclass.c_instruction.**Instruction**(*identifier*, *command*, *executor*, *payloads=None*, *sleep=0*, *timeout=60*)
Bases: *app.utility.base_object.BaseObject*

display

app.objects.secondclass.c_link module

class app.objects.secondclass.c_link.**Link**(*operation*, *command*, *paw*, *ability*, *status=-3*, *score=0*, *jitter=0*, *cleanup=0*, *id=None*, *pin=0*, *host=None*)

Bases: *app.utility.base_object.BaseObject*

apply_id(*host*)

can_ignore()

display

classmethod from_json(*json*)

parse(*operation*, *result*)

pin

states

unique

app.objects.secondclass.c_parser module

class app.objects.secondclass.c_parser.**Parser**(*module*, *parserconfigs*)

Bases: *app.utility.base_object.BaseObject*

display

classmethod from_json(*json*)

unique

app.objects.secondclass.c_parserconfig module

class app.objects.secondclass.c_parserconfig.**ParserConfig**(*source*, *edge=None*, *target=None*, ***kwargs*)

Bases: *app.utility.base_object.BaseObject*

display

classmethod from_json(*json*)

exception app.objects.secondclass.c_parserconfig.**ParserConfigException**

Bases: Exception

app.objects.secondclass.c_relationship module

class app.objects.secondclass.c_relationship.**Relationship**(*source*, *edge=None*, *target=None*, *score=1*)

Bases: *app.utility.base_object.BaseObject*

display

classmethod from_json(*json*)

unique

app.objects.secondclass.c_requirement module

class app.objects.secondclass.c_requirement.**Requirement**(*module*, *relationships*)
Bases: *app.utility.base_object.BaseObject*

display

classmethod from_json(*json*)

unique

app.objects.secondclass.c_result module

class app.objects.secondclass.c_result.**Result**(*id*, *output*, *pid=0*, *status=0*)
Bases: *app.utility.base_object.BaseObject*

app.objects.secondclass.c_rule module

class app.objects.secondclass.c_rule.**Rule**(*action*, *trait*, *match='.*'*)
Bases: *app.utility.base_object.BaseObject*

display

app.objects.secondclass.c_variation module

class app.objects.secondclass.c_variation.**Variation**(*description*, *command*)
Bases: *app.utility.base_object.BaseObject*

command

display

app.objects.secondclass.c_visibility module

class app.objects.secondclass.c_visibility.**Visibility**
Bases: *app.utility.base_object.BaseObject*

MAX_SCORE = 100

MIN_SCORE = 1

apply(*adjustment*)

display

score

15.4.2 Submodules

15.4.3 app.objects.c_ability module

class app.objects.c_ability.**Ability**(*ability_id*, *tactic=None*, *technique_id=None*, *technique=None*, *name=None*, *test=None*, *description=None*, *cleanup=None*, *executor=None*, *platform=None*, *payloads=None*, *parsers=None*, *requirements=None*, *privilege=None*, *timeout=60*, *repeatable=False*, *access=None*, *variations=None*)

Bases: *app.utility.base_object.BaseObject*

RESERVED = {'payload': '#{payload}'}

display

classmethod from_json(*json*)

replace_cleanup(*encoded_cmd*, *payload*)

store(*ram*)

test

unique

which_plugin()

15.4.4 app.objects.c_adversary module

class app.objects.c_adversary.**Adversary**(*adversary_id*, *name*, *description*, *atomic_ordering*)

Bases: *app.utility.base_object.BaseObject*

display

has_ability(*ability*)

store(*ram*)

unique

which_plugin()

15.4.5 app.objects.c_agent module

class app.objects.c_agent.**Agent**(*sleep_min*, *sleep_max*, *watchdog*, *platform='unknown'*, *server='unknown'*, *host='unknown'*, *username='unknown'*, *architecture='unknown'*, *group='red'*, *location='unknown'*, *pid=0*, *ppid=0*, *trusted=True*, *executors=()*, *privilege='User'*, *exe_name='unknown'*, *contact='unknown'*, *paw=None*)

Bases: *app.utility.base_object.BaseObject*

class AgentSchema(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict[KT, VT] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: str = None*)

Bases: marshmallow.schema.Schema

opts = <marshmallow.schema.SchemaOpts object>

remove_nulls(*in_data*, **_)

RESERVED = {'agent_paw': '#{paw}', 'exe_name': '#{exe_name}', 'group': '#{group}',

all_facts()

bootstrap(*data_svc*)

calculate_sleep()

capabilities(*ability_set*)

display

display_name

classmethod from_dict(*dict_obj*)
Creates an Agent object from parameters stored in a dict. AgentSchema is used to validate inputs.

gui_modification(***kwargs*)

heartbeat_modification(***kwargs*)

kill()

privileged_to_run(*ability*)

replace(*encoded_cmd*, *file_svc*)

store(*ram*)

task(*abilities*, *facts=()*)

unique

15.4.6 app.objects.c_obfuscator module

class app.objects.c_obfuscator.**Obfuscator**(*name*, *description*, *module*)
Bases: *app.utility.base_object.BaseObject*

display

load(*agent*)

store(*ram*)

unique

15.4.7 app.objects.c_operation module

class app.objects.c_operation.**Operation**(*name*, *agents*, *adversary*, *id=None*, *jitter='2/8'*, *source=None*, *planner=None*, *state='running'*, *autonomous=True*, *atomic_enabled=False*, *obfuscator='plain-text'*, *group=None*, *auto_close=True*, *visibility=50*, *access=None*)
Bases: *app.utility.base_object.BaseObject*

class Reason
Bases: enum.Enum

An enumeration.

EXECUTOR = 1

FACT_DEPENDENCY = 2

OP_RUNNING = 4

PLATFORM = 0

PRIVILEGE = 3

UNTRUSTED = 5

active_agents()

add_link(*link*)

all_facts()

all_relationships()

apply(*link*)

close()

display

get_active_agent_by_paw(*paw*)

has_fact(*trait*, *value*)

is_closeable()

is_finished()

link_status()

report(*file_svc*, *output=False*, *redacted=False*)

run(*services*)

set_start_details()

states

store(*ram*)

unique

update_operation(*services*)

wait_for_completion()

wait_for_links_completion(*link_ids*)
Wait for started links to be completed :param link_ids: :return: None

15.4.8 app.objects.c_planner module

class app.objects.c_planner.**Planner**(*planner_id*, *name*, *module*, *params*, *stopping_conditions=None*, *description=None*, *ignore_enforcement_modules=()*)
Bases: *app.utility.base_object.BaseObject*

display

store(*ram*)

unique

which_plugin()

15.4.9 app.objects.c_plugin module

class app.objects.c_plugin.**Plugin**(*name='virtual'*, *description=None*, *address=None*, *enabled=False*, *data_dir=None*, *access=None*)

Bases: *app.utility.base_object.BaseObject*

destroy(*services*)

display

enable(*services*)

load()

store(*ram*)

unique

15.4.10 app.objects.c_schedule module

class app.objects.c_schedule.**Schedule**(*name*, *schedule*, *task*)

Bases: *app.utility.base_object.BaseObject*

display

store(*ram*)

unique

15.4.11 app.objects.c_source module

class app.objects.c_source.**Source**(*identifier*, *name*, *facts*, *rules=()*, *adjustments=()*)

Bases: *app.utility.base_object.BaseObject*

display

store(*ram*)

unique

15.5 app.service namespace

15.5.1 Submodules

15.5.2 app.service.app_svc module

class app.service.app_svc.**AppService**(*application*)

Bases: *app.utility.base_service.BaseService*

find_link(*unique*)

Locate a given link by its unique property

Parameters **unique** –

Returns

load_plugins()
Store all plugins in the data store

Returns

register_contacts()

resume_operations()
Resume all unfinished operations

Returns None

retrieve_compiled_file(*name*, *platform*)

run_scheduler()
Kick off all scheduled jobs, as their schedule determines

Returns

start_sniffer_untrusted_agents()
Cyclic function that repeatedly checks if there are agents to be marked as untrusted

Returns None

teardown()

15.5.3 app.service.auth_svc module

class app.service.auth_svc.**AuthService**
Bases: *app.utility.base_service.BaseService*

class User(*username*, *password*, *permissions*)
Bases: tuple

password
Alias for field number 1

permissions
Alias for field number 2

username
Alias for field number 0

apply(*app*, *users*)
Set up security on server boot :param app: :param users: :return: None

check_permissions(*group*, *request*)
Check if a request is allowed based on the user permissions :param request: :return: None

get_permissions(*request*)

login_user(*request*)
Log a user in and save the session :param request: :return: the response/location of where the user is trying to navigate

static logout_user(*request*)
Log the user out :param request: :return: None

class app.service.auth_svc.**DictionaryAuthorizationPolicy**(*user_map*)
Bases: aiohttp_security.abc.AbstractAuthorizationPolicy

authorized_userid(*identity*)
Retrieve authorized user id. Return the user_id of the user identified by the identity or 'None' if no user exists related to the identity.

permits(*identity*, *permission*, *context=None*)
Check user permissions. Return True if the identity is allowed the permission in the current context, else return False.

app.service.auth_svc.**check_authorization**(*func*)
Authorization Decorator This requires that the calling class have *self.auth_svc* set to the authentication service.

15.5.4 app.service.contact_svc module

class app.service.contact_svc.**ContactService**
Bases: *app.utility.base_service.BaseService*

build_filename()

handle_heartbeat(***kwargs*)

register(*contact*)

app.service.contact_svc.**report**(*func*)

15.5.5 app.service.data_svc module

class app.service.data_svc.**Adjustment**(*ability_id*, *trait*, *value*, *offset*)
Bases: tuple

ability_id
Alias for field number 0

offset
Alias for field number 3

trait
Alias for field number 1

value
Alias for field number 2

class app.service.data_svc.**DataService**
Bases: *app.utility.base_service.BaseService*

apply(*collection*)
Add a new collection to RAM

Parameters **collection** –

Returns

static destroy()
Clear out all data

Returns

load_data(*plugins=()*)
Non-blocking read all the data sources to populate the object store

Returns None

locate(*object_name*, *match=None*)
Find all c_objects which match a search. Return all c_objects if no match.

Parameters

- **object_name** –
- **match** – dict()

Returns a list of c_object types

reload_data(*plugins=()*)
Blocking read all the data sources to populate the object store

Returns None

remove(*object_name*, *match*)
Remove any c_objects which match a search

Parameters

- **object_name** –
- **match** – dict()

Returns

restore_state()
Restore the object database

Returns

save_state()
Save RAM database to file

Returns

store(*c_object*)
Accept any c_object type and store it (create/update) in RAM

Parameters **c_object** –

Returns a single c_object

15.5.6 app.service.file_svc module

class app.service.file_svc.**FileSvc**
Bases: *app.utility.base_service.BaseService*

add_special_payload(*name*, *func*)
Call a special function when specific payloads are downloaded

Parameters

- **name** –
- **func** –

Returns

compile_go(*platform*, *output*, *src_fle*, *arch='amd64'*, *ldflags='-s -w'*, *cflags=''*, *buildmode=''*, *build_dir='.'*)
Dynamically compile a go file

Parameters

- **platform** –
- **output** –
- **src_fle** –

- **arch** – Compile architecture selection (defaults to AMD64)
- **ldflags** – A string of ldflags to use when building the go executable
- **cflags** – A string of CFLAGS to pass to the go compiler
- **buildmode** – GO compiler buildmode flag
- **build_dir** – The path to build should take place in

Returns

create_exfil_sub_directory(*dir_name*)

find_file_path(*name*, *location=''*)
Find the location on disk of a file by name.

Parameters

- **name** –
- **location** –

Returns a tuple: the plugin the file is found in & the relative file path

get_file(*headers*)
Retrieve file :param headers: headers dictionary. The *file* key is REQUIRED. :type headers: dict or dict-equivalent :return: File contents and optionally a display_name if the payload is a special payload :raises: KeyError if file key is not provided, FileNotFoundError if file cannot be found

get_payload_name_from_uuid(*payload*)

read_file(*name*, *location='payloads'*)
Open a file and read the contents

Parameters

- **name** –
- **location** –

Returns a tuple (file_path, contents)

read_result_file(*link_id*, *location='data/results'*)
Read a result file. If file encryption is enabled, this method will return the plaintext content.

Parameters

- **link_id** – The id of the link to return results from.
- **location** – The path to results directory.

Returns

save_file(*filename*, *payload*, *target_dir*)

save_multipart_file_upload(*request*, *target_dir*)
Accept a multipart file via HTTP and save it to the server

Parameters

- **request** –
- **target_dir** – The path of the directory to save the uploaded file to.

write_result_file(*link_id*, *output*, *location='data/results'*)
Writes the results of a link execution to disk. If file encryption is enabled, the results file will contain ciphertext.

Parameters

- **link_id** – The link id of the result being written.
- **output** – The content of the link's output.
- **location** – The path to the results directory.

Returns

15.5.7 app.service.learning_svc module

class app.service.learning_svc.**LearningService**
Bases: *app.utility.base_service.BaseService*

static add_parsers(*directory*)

build_model()
The model is a static set of all variables used inside all ability commands This can be used to determine which facts - when found together - are more likely to be used together :return:

learn(*facts*, *link*, *blob*)

15.5.8 app.service.planning_svc module

class app.service.planning_svc.**PlanningService**
Bases: *app.utility.base_planning_svc.BasePlanningService*

generate_and_trim_links(*agent*, *operation*, *abilities*, *trim=True*)
repeated subroutine

get_cleanup_links(*operation*, *agent=None*)
For a given operation, create all cleanup links. If agent is supplied, only return cleanup links for that agent.

Parameters

- **operation** –
- **agent** –

Returns None

get_links(*operation*, *agent=None*, *trim=True*, *planner=None*, *stopping_conditions=None*)
For an operation and agent combination, create links (that can be executed). When no agent is supplied, links for all agents are returned

Parameters

- **operation** –
- **agent** –
- **trim** – call trim_links() on list of links before returning
- **planner** –
- **stopping_conditions** –

Returns a list of links

static sort_links(*links*)
Sort links by their score then by the order they are defined in an adversary profile

15.5.9 app.service.rest_svc module

class app.service.rest_svc.**RestService**
Bases: *app.utility.base_service.BaseService*

apply_potential_link(*link*)

construct_agents_for_group(*group*)

create_operation(*access*, *data*)

create_schedule(*access*, *data*)

delete_ability(*data*)

delete_adversary(*data*)

delete_agent(*data*)

delete_operation(*data*)

display_objects(*object_name*, *data*)

display_operation_report(*data*)

display_result(*data*)

download_contact_report(*contact*)

find_abilities(*paw*)

get_link_pin(*json_data*)

get_potential_links(*op_id*, *paw=None*)

list_payloads()

persist_ability(*data*)

persist_adversary(*data*)
Save a new adversary from either the GUI or REST API. This writes a new YML file into the core data/ directory.

Parameters **data** –

Returns the ID of the created adversary

persist_source(*data*)

task_agent_with_ability(*paw*, *ability_id*, *facts=()*)

update_agent_data(*data*)

update_chain_data(*data*)

update_config(*data*)

update_operation(*op_id*, *state=None*, *autonomous=None*)

update_planner(*data*)
Update a new planner from either the GUI or REST API with new stopping conditions. This overwrites the existing YML file.

Parameters **data** –

Returns the ID of the created adversary

15.6 app.utility namespace

15.6.1 Submodules

15.6.2 app.utility.base_obfuscator module

class app.utility.base_obfuscator.**BaseObfuscator**(*agent*)
Bases: *app.utility.base_world.BaseWorld*

run(*link*, ***kwargs*)

15.6.3 app.utility.base_object module

class app.utility.base_object.**BaseObject**
Bases: *app.utility.base_world.BaseWorld*

access

static clean(*d*)

static hash(*s*)

match(*criteria*)

replace_app_props(*encoded_string*)

static retrieve(*collection*, *unique*)

update(*field*, *value*)

15.6.4 app.utility.base_parser module

class app.utility.base_parser.**BaseParser**(*parser_info*)
Bases: object

static broadcastip(*blob*)

static email(*blob*)
Parse out email addresses :param blob: :return:

static filename(*blob*)
Parse out filenames :param blob: :return:

static ip(*blob*)

static line(*blob*)
Split a blob by line :param blob: :return:

static load_json(*blob*)

static set_value(*search*, *match*, *used_facts*)
Determine the value of a source/target for a Relationship :param search: a fact property to look for; either a source or target fact :param match: a parsing match :param used_facts: a list of facts that were used in a command :return: either None, the value of a matched used_fact, or the parsing match

15.6.5 app.utility.base_planning_svc module

class app.utility.base_planning_svc.**BasePlanningService**
Bases: *app.utility.base_service.BaseService*

add_test_variants(*links*, *agent*, *facts=()*, *rules=()*)
Create a list of all possible links for a given set of templates

Parameters

- **links** –
- **agent** –
- **facts** –
- **rules** –

Returns updated list of links

obfuscate_commands(*agent*, *obfuscator*, *links*)

re_index = re.compile('(?<=\\[filters\\().+?(?=\\)\\])')

re_limited = re.compile('#{.*\\[*\\]}')

re_trait = re.compile('(?<=\\{).+?(?=\\[)')

re_variable = re.compile('#{(.*?)}', re.DOTALL)

static remove_completed_links(*operation*, *agent*, *links*)
Remove any links that have already been completed by the operation for the agent

Parameters

- **operation** –
- **links** –
- **agent** –

Returns updated list of links

static remove_links_above_visibility(*links*, *operation*)

static remove_links_missing_facts(*links*)
Remove any links that did not have facts encoded into command

Parameters **links** –

Returns updated list of links

remove_links_missing_requirements(*links*, *operation*)

trim_links(*operation*, *links*, *agent*)

Trim links in supplied list. Where 'trim' entails:

- adding all possible test variants
- removing completed links (i.e. agent has already completed)
- removing links that did not have template fact variables replaced by fact values

Parameters

- **operation** –
- **links** –

- **agent** –

Returns trimmed list of links

15.6.6 app.utility.base_service module

class app.utility.base_service.**BaseService**
Bases: *app.utility.base_world.BaseWorld*

add_service(*name*, *svc*)

classmethod get_service(*name*)

classmethod get_services()

15.6.7 app.utility.base_world module

class app.utility.base_world.**BaseWorld**
Bases: object

A collection of base static functions for service & object module usage

class Access
Bases: enum.Enum

An enumeration.

APP = 0

BLUE = 2

RED = 1

class Privileges
Bases: enum.Enum

An enumeration.

Elevated = 1

User = 0

static apply_config(*name*, *config*)

static create_logger(*name*)

static decode_bytes(*s*)

static encode_string(*s*)

static generate_name(*size=16*)

static generate_number(*size=6*)

static get_config(*prop=None*, *name=None*)

static get_current_timestamp(*date_format='%Y-%m-%d %H:%M:%S'*)

static is_base64(*s*)

static is_uuid4(*s*)

static jitter(*fraction*)

static load_module(*module_type*, *module_info*)

static prepend_to_file(*filename*, *line*)

re_base64 = re.compile('[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}',

static set_config(*name*, *prop*, *value*)

static strip_yml(*path*)

static walk_file_path(*path*, *target*)

15.6.8 app.utility.file_decryptor module

app.utility.file_decryptor.**decrypt**(*filename*, *configuration*, *output_file=None*, *b64decode=False*)

app.utility.file_decryptor.**get_encryptor**(*salt*, *key*)

app.utility.file_decryptor.**read**(*filename*, *encryptor*)

15.6.9 app.utility.payload_encoder module

This module contains helper functions for encoding and decoding payload files.

If AV is running on the server host, then it may sometimes flag, quarantine, or delete CALDERA payloads. To help prevent this, encoded payloads can be used to prevent AV from breaking the server. The convention expected by the server is that encoded payloads will be XOR'ed with the DEFAULT_KEY contained in the payload_encoder.py module.

Additionally, payload_encoder.py can be used from the command-line to add a new encoded payload.

```
` python /path/to/payload_encoder.py input_file output_file `
```

NOTE: In order for the server to detect the availability of an encoded payload, the payload file's name must end in the *.xored* extension.

app.utility.payload_encoder.**xor_bytes**(*in_bytes*, *key=None*)

app.utility.payload_encoder.**xor_file**(*input_file*, *output_file=None*, *key=None*)

15.6.10 app.utility.rule_set module

class app.utility.rule_set.**RuleAction**

Bases: enum.Enum

An enumeration.

ALLOW = 1

DENY = 0

class app.utility.rule_set.**RuleSet**(*rules*)

Bases: object

apply_rules(*facts*)

is_fact_allowed(*fact*)

CHAPTER 16

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

app.api.packs.advanced, 45
app.api.packs.campaign, 45
app.api.rest_api, 45
app.contacts.contact_gist, 46
app.contacts.contact_html, 47
app.contacts.contact_http, 47
app.contacts.contact_tcp, 47
app.contacts.contact_udp, 47
app.contacts.contact_websocket, 47
app.contacts.handles.h_beacon, 46
app.learning.p_ip, 48
app.learning.p_path, 48
app.objects.c_ability, 51
app.objects.c_adversary, 51
app.objects.c_agent, 51
app.objects.c_obfuscator, 52
app.objects.c_operation, 52
app.objects.c_planner, 53
app.objects.c_plugin, 54
app.objects.c_schedule, 54
app.objects.c_source, 54
app.objects.secondclass.c_fact, 48
app.objects.secondclass.c_instruction, 48
app.objects.secondclass.c_link, 49
app.objects.secondclass.c_parser, 49
app.objects.secondclass.c_parserconfig, 49
app.objects.secondclass.c_relationship, 49
app.objects.secondclass.c_requirement, 50
app.objects.secondclass.c_result, 50
app.objects.secondclass.c_rule, 50
app.objects.secondclass.c_variation, 50
app.objects.secondclass.c_visibility, 50
app.service.app_svc, 54
app.service.auth_svc, 55
app.service.contact_svc, 56
app.service.data_svc, 56
app.service.file_svc, 57
app.service.learning_svc, 59
app.service.planning_svc, 59
app.service.rest_svc, 60
app.utility.base_obfuscator, 61
app.utility.base_object, 61
app.utility.base_parser, 61
app.utility.base_planning_svc, 62
app.utility.base_service, 63
app.utility.base_world, 63
app.utility.file_decryptor, 64
app.utility.payload_encoder, 64
app.utility.rule_set, 64

Index

A

Ability (*class in app.objects.c_ability*), 51
ability_id (*app.service.data_svc.Adjustment attribute*), 56
accept() (*app.contacts.contact_tcp.TcpSessionHandler method*), 47
access (*app.utility.base_object.BaseObject attribute*), 61
active_agents() (*app.objects.c_operation.Operation method*), 53
add_link() (*app.objects.c_operation.Operation method*), 53
add_parsers() (*app.service.learning_svc.LearningService static method*), 59
add_service() (*app.utility.base_service.BaseService method*), 63
add_special_payload() (*app.service.file_svc.FileSvc method*), 57
add_test_variants() (*app.utility.base_planning_svc.BasePlanningService method*), 62
Adjustment (*class in app.service.data_svc*), 56
AdvancedPack (*class in app.api.packs.advanced*), 45
Adversary (*class in app.objects.c_adversary*), 51
Agent (*class in app.objects.c_agent*), 51
Agent.AgentSchema (*class in app.objects.c_agent*), 51
all_facts() (*app.objects.c_agent.Agent method*), 52
all_facts() (*app.objects.c_operation.Operation method*), 53
all_relationships() (*app.objects.c_operation.Operation method*), 53
ALLOW (*app.utility.rule_set.RuleAction attribute*), 64
api_access() (*in module app.contacts.contact_gist*), 46
APP (*app.utility.base_world.BaseWorld.Access attribute*), 63
app.api.packs.advanced (*module*), 45
app.api.packs.campaign (*module*), 45
app.api.rest_api (*module*), 45
app.contacts.contact_gist (*module*), 46
app.contacts.contact_html (*module*), 47
app.contacts.contact_http (*module*), 47
app.contacts.contact_tcp (*module*), 47
app.contacts.contact_udp (*module*), 47
app.contacts.contact_websocket (*module*), 47
app.contacts.handles.h_beacon (*module*), 46
app.learning.p_ip (*module*), 48
app.learning.p_path (*module*), 48
app.objects.c_ability (*module*), 51
app.objects.c_adversary (*module*), 51
app.objects.c_agent (*module*), 51
app.objects.c_obfuscator (*module*), 52
app.objects.c_operation (*module*), 52
app.objects.c_planner (*module*), 53
app.objects.c_plugin (*module*), 54
app.objects.c_schedule (*module*), 54
app.objects.c_source (*module*), 54
app.objects.secondclass.c_fact (*module*), 48
app.objects.secondclass.c_instruction (*module*), 48
app.objects.secondclass.c_link (*module*), 49
app.objects.secondclass.c_parser (*module*), 49
app.objects.secondclass.c_parserconfig (*module*), 49
app.objects.secondclass.c_relationship (*module*), 49
app.objects.secondclass.c_requirement (*module*), 50
app.objects.secondclass.c_result (*module*), 50
app.objects.secondclass.c_rule (*module*), 50
app.objects.secondclass.c_variation

(*module*), 50
app.objects.secondclass.c_visibility (*module*), 50
app.service.app_svc (*module*), 54
app.service.auth_svc (*module*), 55
app.service.contact_svc (*module*), 56
app.service.data_svc (*module*), 56
app.service.file_svc (*module*), 57
app.service.learning_svc (*module*), 59
app.service.planning_svc (*module*), 59
app.service.rest_svc (*module*), 60
app.utility.base_obfuscator (*module*), 61
app.utility.base_object (*module*), 61
app.utility.base_parser (*module*), 61
app.utility.base_planning_svc (*module*), 62
app.utility.base_service (*module*), 63
app.utility.base_world (*module*), 63
app.utility.file_decryptor (*module*), 64
app.utility.payload_encoder (*module*), 64
app.utility.rule_set (*module*), 64
apply() (*app.objects.c_operation.Operation method*), 53
apply() (*app.objects.secondclass.c_visibility.Visibility method*), 50
apply() (*app.service.auth_svc.AuthService method*), 55
apply() (*app.service.data_svc.DataService method*), 56
apply_config() (*app.utility.base_world.BaseWorld static method*), 63
apply_id() (*app.objects.secondclass.c_link.Link method*), 49
apply_potential_link() (*app.service.rest_svc.RestService method*), 60
apply_rules() (*app.utility.rule_set.RuleSet method*), 64
AppService (*class in app.service.app_svc*), 54
authorized_userid() (*app.service.auth_svc.DictionaryAuthorizationPolicy method*), 55
AuthService (*class in app.service.auth_svc*), 55
AuthService.User (*class in app.service.auth_svc*), 55

B

BaseObfuscator (*class in app.utility.base_obfuscator*), 61
BaseObject (*class in app.utility.base_object*), 61
BaseParser (*class in app.utility.base_parser*), 61
BasePlanningService (*class in app.utility.base_planning_svc*), 62
BaseService (*class in app.utility.base_service*), 63
BaseWorld (*class in app.utility.base_world*), 63
BaseWorld.Access (*class in app.utility.base_world*), 63
BaseWorld.Privileges (*class in app.utility.base_world*), 63
BLUE (*app.utility.base_world.BaseWorld.Access attribute*), 63
bootstrap() (*app.objects.c_agent.Agent method*), 52
broadcastip() (*app.utility.base_parser.BaseParser static method*), 61
build_filename() (*app.service.contact_svc.ContactService method*), 56
build_model() (*app.service.learning_svc.LearningService method*), 59

C

calculate_sleep() (*app.objects.c_agent.Agent method*), 52
CampaignPack (*class in app.api.packs.campaign*), 45
can_ignore() (*app.objects.secondclass.c_link.Link method*), 49
capabilities() (*app.objects.c_agent.Agent method*), 52
check_authorization() (*in module app.service.auth_svc*), 56
check_permissions() (*app.service.auth_svc.AuthService method*), 55
clean() (*app.utility.base_object.BaseObject static method*), 61
close() (*app.objects.c_operation.Operation method*), 53
command (*app.objects.secondclass.c_variation.Variation attribute*), 50
compile_go() (*app.service.file_svc.FileSvc method*), 57
construct_agents_for_group() (*app.service.rest_svc.RestService method*), 60
ContactService (*class in app.service.contact_svc*), 56
create_exfil_sub_directory() (*app.service.file_svc.FileSvc method*), 58
create_logger() (*app.utility.base_world.BaseWorld static method*), 63
create_operation() (*app.service.rest_svc.RestService method*), 60
create_schedule() (*app.service.rest_svc.RestService method*), 60

D

datagram_received() (*app.contacts.contact_udp.Handler method*), 47

DataService (*class in app.service.data_svc*), 56
decode_bytes() (*app.utility.base_world.BaseWorld static method*), 63
decrypt() (*in module app.utility.file_decryptor*), 64
delete_ability() (*app.service.rest_svc.RestService method*), 60
delete_adversary() (*app.service.rest_svc.RestService method*), 60
delete_agent() (*app.service.rest_svc.RestService method*), 60
delete_operation() (*app.service.rest_svc.RestService method*), 60
DENY (*app.utility.rule_set.RuleAction attribute*), 64
destroy() (*app.objects.c_plugin.Plugin method*), 54
destroy() (*app.service.data_svc.DataService static method*), 56
DictionaryAuthorizationPolicy (*class in app.service.auth_svc*), 55
display (*app.objects.c_ability.Ability attribute*), 51
display (*app.objects.c_adversary.Adversary attribute*), 51
display (*app.objects.c_agent.Agent attribute*), 52
display (*app.objects.c_obfuscator.Obfuscator attribute*), 52
display (*app.objects.c_operation.Operation attribute*), 53
display (*app.objects.c_planner.Planner attribute*), 53
display (*app.objects.c_plugin.Plugin attribute*), 54
display (*app.objects.c_schedule.Schedule attribute*), 54
display (*app.objects.c_source.Source attribute*), 54
display (*app.objects.secondclass.c_fact.Fact attribute*), 48
display (*app.objects.secondclass.c_instruction.Instruction attribute*), 48
display (*app.objects.secondclass.c_link.Link attribute*), 49
display (*app.objects.secondclass.c_parser.Parser attribute*), 49
display (*app.objects.secondclass.c_parserconfig.ParserConfig attribute*), 49
display (*app.objects.secondclass.c_relationship.Relationship attribute*), 49
display (*app.objects.secondclass.c_requirement.Requirement attribute*), 50
display (*app.objects.secondclass.c_rule.Rule attribute*), 50
display (*app.objects.secondclass.c_variation.Variation attribute*), 50
display (*app.objects.secondclass.c_visibility.Visibility attribute*), 50
display_name (*app.objects.c_agent.Agent attribute*), 52
display_objects() (*app.service.rest_svc.RestService method*), 60
display_operation_report() (*app.service.rest_svc.RestService method*), 60
display_result() (*app.service.rest_svc.RestService method*), 60
download_contact_report() (*app.service.rest_svc.RestService method*), 60
download_file() (*app.api.rest_api.RestApi method*), 45

E

Elevated (*app.utility.base_world.BaseWorld.Privileges attribute*), 63
email() (*app.utility.base_parser.BaseParser static method*), 61
enable() (*app.api.packs.advanced.AdvancedPack method*), 45
enable() (*app.api.packs.campaign.CampaignPack method*), 45
enable() (*app.api.rest_api.RestApi method*), 46
enable() (*app.objects.c_plugin.Plugin method*), 54
encode_string() (*app.utility.base_world.BaseWorld static method*), 63
escaped() (*app.objects.secondclass.c_fact.Fact method*), 48
EXECUTOR (*app.objects.c_operation.Operation.Reason attribute*), 52

F

Fact (*class in app.objects.secondclass.c_fact*), 48
FACT_DEPENDENCY (*app.objects.c_operation.Operation.Reason attribute*), 53
filename() (*app.utility.base_parser.BaseParser static method*), 61
FileSvc (*class in app.service.file_svc*), 57
find_abilities() (*app.service.rest_svc.RestService method*), 60
find_file_path() (*app.service.file_svc.FileSvc method*), 58
find_link() (*app.service.app_svc.AppService method*), 54
from_dict() (*app.objects.c_agent.Agent class method*), 52
from_json() (*app.objects.c_ability.Ability class method*), 51
from_json() (*app.objects.secondclass.c_link.Link class method*), 49
from_json() (*app.objects.secondclass.c_parser.Parser class method*), 49

from_json() (app.objects.secondclass.c_parserconfig.ParserConfig class method), 49
from_json() (app.objects.secondclass.c_relationship.Relationship class method), 49
from_json() (app.objects.secondclass.c_requirement.Requirement class method), 50

G

generate_and_trim_links() (app.service.planning_svc.PlanningService method), 59
generate_name() (app.utility.base_world.BaseWorld static method), 63
generate_number() (app.utility.base_world.BaseWorld static method), 63
get_active_agent_by_paw() (app.objects.c_operation.Operation method), 53
get_beacons() (app.contacts.contact_gist.Gist method), 46
get_cleanup_links() (app.service.planning_svc.PlanningService method), 59
get_config() (app.utility.base_world.BaseWorld static method), 63
get_current_timestamp() (app.utility.base_world.BaseWorld static method), 63
get_encryptor() (in module app.utility.file_decryptor), 64
get_file() (app.service.file_svc.FileSvc method), 58
get_link_pin() (app.service.rest_svc.RestService method), 60
get_links() (app.service.planning_svc.PlanningService method), 59
get_payload_name_from_uuid() (app.service.file_svc.FileSvc method), 58
get_permissions() (app.service.auth_svc.AuthService method), 55
get_potential_links() (app.service.rest_svc.RestService method), 60
get_results() (app.contacts.contact_gist.Gist method), 46
get_service() (app.utility.base_service.BaseService class method), 63
get_services() (app.utility.base_service.BaseService class method), 63
Gist (class in app.contacts.contact_gist), 46
gist_operation_loop() (app.contacts.contact_gist.Gist method), 46
gui_modification() (app.objects.c_agent.Agent method), 52

H

handle() (class in app.contacts.handles.h_beacon), 46
handle() (app.contacts.contact_websocket.Handler method), 47
handle_beacons() (app.contacts.contact_gist.Gist method), 46
handle_heartbeat() (app.service.contact_svc.ContactService method), 56
Handler (class in app.contacts.contact_udp), 47
Handler (class in app.contacts.contact_websocket), 47
has_ability() (app.objects.c_adversary.Adversary method), 51
has_fact() (app.objects.c_operation.Operation method), 53
hash() (app.utility.base_object.BaseObject static method), 61
heartbeat_modification() (app.objects.c_agent.Agent method), 52
Html (class in app.contacts.contact_html), 47
Http (class in app.contacts.contact_http), 47

I

Instruction (class in app.objects.secondclass.c_instruction), 48
ip() (app.utility.base_parser.BaseParser static method), 61
is_base64() (app.utility.base_world.BaseWorld static method), 63
is_closeable() (app.objects.c_operation.Operation method), 53
is_fact_allowed() (app.utility.rule_set.RuleSet method), 64
is_finished() (app.objects.c_operation.Operation method), 53
is_uuid4() (app.utility.base_world.BaseWorld static method), 63

J

jitter() (app.utility.base_world.BaseWorld static method), 63

K

kill() (app.objects.c_agent.Agent method), 52

L

landing() (app.api.rest_api.RestApi method), 46
learn() (app.service.learning_svc.LearningService method), 59
LearningService (class in app.service.learning_svc), 59

line() (*app.utility.base_parser.BaseParser static method*), 61
Link (*class in app.objects.secondclass.c_link*), 49
link_status() (*app.objects.c_operation.Operation method*), 53
list_payloads() (*app.service.rest_svc.RestService method*), 60
load() (*app.objects.c_obfuscator.Obfuscator method*), 52
load() (*app.objects.c_plugin.Plugin method*), 54
load_data() (*app.service.data_svc.DataService method*), 56
load_json() (*app.utility.base_parser.BaseParser static method*), 61
load_module() (*app.utility.base_world.BaseWorld static method*), 63
load_plugins() (*app.service.app_svc.AppService method*), 54
locate() (*app.service.data_svc.DataService method*), 56
login() (*app.api.rest_api.RestApi method*), 46
login_user() (*app.service.auth_svc.AuthService method*), 55
logout() (*app.api.rest_api.RestApi method*), 46
logout_user() (*app.service.auth_svc.AuthService static method*), 55

M

match() (*app.utility.base_object.BaseObject method*), 61
MAX_SCORE (*app.objects.secondclass.c_visibility.Visibility attribute*), 50
MIN_SCORE (*app.objects.secondclass.c_visibility.Visibility attribute*), 50

O

obfuscate_commands() (*app.utility.base_planning_svc.BasePlanningService method*), 62
Obfuscator (*class in app.objects.c_obfuscator*), 52
offset (*app.service.data_svc.Adjustment attribute*), 56
OP_RUNNING (*app.objects.c_operation.Operation.Reason attribute*), 53
Operation (*class in app.objects.c_operation*), 52
Operation.Reason (*class in app.objects.c_operation*), 52
operation_loop() (*app.contacts.contact_tcp.Tcp method*), 47
opts (*app.objects.c_agent.Agent.AgentSchema attribute*), 51

P

parse() (*app.learning.p_ip.Parser method*), 48
parse() (*app.learning.p_path.Parser method*), 48
parse() (*app.objects.secondclass.c_link.Link method*), 49
Parser (*class in app.learning.p_ip*), 48
Parser (*class in app.learning.p_path*), 48
Parser (*class in app.objects.secondclass.c_parser*), 49
ParserConfig (*class in app.objects.secondclass.c_parserconfig*), 49
ParserConfigException, 49
password (*app.service.auth_svc.AuthService.User attribute*), 55
permissions (*app.service.auth_svc.AuthService.User attribute*), 55
permits() (*app.service.auth_svc.DictionaryAuthorizationPolicy method*), 56
persist_ability() (*app.service.rest_svc.RestService method*), 60
persist_adversary() (*app.service.rest_svc.RestService method*), 60
persist_source() (*app.service.rest_svc.RestService method*), 60
pin (*app.objects.secondclass.c_link.Link attribute*), 49
Planner (*class in app.objects.c_planner*), 53
PlanningService (*class in app.service.planning_svc*), 59
PLATFORM (*app.objects.c_operation.Operation.Reason attribute*), 53
Plugin (*class in app.objects.c_plugin*), 54
prepend_to_file() (*app.utility.base_world.BaseWorld static method*), 63
PRIVILEGE (*app.objects.c_operation.Operation.Reason attribute*), 53
privileged_to_run() (*app.objects.c_agent.Agent method*), 52

R

re_base64 (*app.utility.base_world.BaseWorld attribute*), 64
re_index (*app.utility.base_planning_svc.BasePlanningService attribute*), 62
re_limited (*app.utility.base_planning_svc.BasePlanningService attribute*), 62
re_trait (*app.utility.base_planning_svc.BasePlanningService attribute*), 62
re_variable (*app.utility.base_planning_svc.BasePlanningService attribute*), 62
read() (*in module app.utility.file_decryptor*), 64
read_file() (*app.service.file_svc.FileSvc method*), 58
read_result_file() (*app.service.file_svc.FileSvc method*), 58

RED (*app.utility.base_world.BaseWorld.Access attribute*), 63
refresh() (*app.contacts.contact_tcp.TcpSessionHandler method*), 47
register() (*app.service.contact_svc.ContactService method*), 56
register_contacts() (*app.service.app_svc.AppService method*), 55
Relationship (*class in app.objects.secondclass.c_relationship*), 49
reload_data() (*app.service.data_svc.DataService method*), 57
remove() (*app.service.data_svc.DataService method*), 57
remove_completed_links() (*app.utility.base_planning_svc.BasePlanningService static method*), 62
remove_links_above_visibility() (*app.utility.base_planning_svc.BasePlanningService static method*), 62
remove_links_missing_facts() (*app.utility.base_planning_svc.BasePlanningService static method*), 62
remove_links_missing_requirements() (*app.utility.base_planning_svc.BasePlanningService method*), 62
remove_nulls() (*app.objects.c_agent.Agent.AgentSchema method*), 52
replace() (*app.objects.c_agent.Agent method*), 52
replace_app_props() (*app.utility.base_object.BaseObject method*), 61
replace_cleanup() (*app.objects.c_ability.Ability method*), 51
report() (*app.objects.c_operation.Operation method*), 53
report() (*in module app.service.contact_svc*), 56
Requirement (*class in app.objects.secondclass.c_requirement*), 50
RESERVED (*app.objects.c_ability.Ability attribute*), 51
RESERVED (*app.objects.c_agent.Agent attribute*), 52
rest_core() (*app.api.rest_api.RestApi method*), 46
RestApi (*class in app.api.rest_api*), 45
restore_state() (*app.service.data_svc.DataService method*), 57
RestService (*class in app.service.rest_svc*), 60
Result (*class in app.objects.secondclass.c_result*), 50
resume_operations() (*app.service.app_svc.AppService method*), 55
retrieve() (*app.utility.base_object.BaseObject static method*), 61
retrieve_compiled_file() (*app.service.app_svc.AppService method*), 55
retrieve_config() (*app.contacts.contact_gist.Gist method*), 46
Rule (*class in app.objects.secondclass.c_rule*), 50
RuleAction (*class in app.utility.rule_set*), 64
RuleSet (*class in app.utility.rule_set*), 64
run() (*app.contacts.handles.h_beacon.Handle static method*), 46
run() (*app.objects.c_operation.Operation method*), 53
run() (*app.utility.base_obfuscator.BaseObfuscator method*), 61
run_scheduler() (*app.service.app_svc.AppService method*), 55

S

save_file() (*app.service.file_svc.FileSvc method*), 58
save_multipart_file_upload() (*app.service.file_svc.FileSvc method*), 58
save_state() (*app.service.data_svc.DataService method*), 57
Schedule (*class in app.objects.c_schedule*), 54
score (*app.objects.secondclass.c_visibility.Visibility attribute*), 50
send() (*app.contacts.contact_tcp.TcpSessionHandler method*), 47
set_config() (*app.utility.base_world.BaseWorld static method*), 64
set_start_details() (*app.objects.c_operation.Operation method*), 53
set_value() (*app.utility.base_parser.BaseParser static method*), 61
sort_links() (*app.service.planning_svc.PlanningService static method*), 59
Source (*class in app.objects.c_source*), 54
start() (*app.contacts.contact_gist.Gist method*), 46
start() (*app.contacts.contact_html.Html method*), 47
start() (*app.contacts.contact_http.Http method*), 47
start() (*app.contacts.contact_tcp.Tcp method*), 47
start() (*app.contacts.contact_udp.Udp method*), 47
start() (*app.contacts.contact_websocket.WebSocket method*), 47
start_sniffer_untrusted_agents() (*app.service.app_svc.AppService method*), 55
states (*app.objects.c_operation.Operation attribute*), 53
states (*app.objects.secondclass.c_link.Link attribute*), 49
store() (*app.objects.c_ability.Ability method*), 51

store() (*app.objects.c_adversary.Adversary method*), 51
store() (*app.objects.c_agent.Agent method*), 52
store() (*app.objects.c_obfuscator.Obfuscator method*), 52
store() (*app.objects.c_operation.Operation method*), 53
store() (*app.objects.c_planner.Planner method*), 53
store() (*app.objects.c_plugin.Plugin method*), 54
store() (*app.objects.c_schedule.Schedule method*), 54
store() (*app.objects.c_source.Source method*), 54
store() (*app.service.data_svc.DataService method*), 57
strip_yml() (*app.utility.base_world.BaseWorld static method*), 64

T

task() (*app.objects.c_agent.Agent method*), 52
task_agent_with_ability() (*app.service.rest_svc.RestService method*), 60
Tcp (*class in app.contacts.contact_tcp*), 47
TcpSessionHandler (*class in app.contacts.contact_tcp*), 47
teardown() (*app.service.app_svc.AppService method*), 55
test (*app.objects.c_ability.Ability attribute*), 51
trait (*app.service.data_svc.Adjustment attribute*), 56
trim_links() (*app.utility.base_planning_svc.BasePlanningService method*), 62

U

Udp (*class in app.contacts.contact_udp*), 47
unique (*app.objects.c_ability.Ability attribute*), 51
unique (*app.objects.c_adversary.Adversary attribute*), 51
unique (*app.objects.c_agent.Agent attribute*), 52
unique (*app.objects.c_obfuscator.Obfuscator attribute*), 52
unique (*app.objects.c_operation.Operation attribute*), 53
unique (*app.objects.c_planner.Planner attribute*), 53
unique (*app.objects.c_plugin.Plugin attribute*), 54
unique (*app.objects.c_schedule.Schedule attribute*), 54
unique (*app.objects.c_source.Source attribute*), 54
unique (*app.objects.secondclass.c_fact.Fact attribute*), 48
unique (*app.objects.secondclass.c_link.Link attribute*), 49
unique (*app.objects.secondclass.c_parser.Parser attribute*), 49
unique (*app.objects.secondclass.c_relationship.Relationship attribute*), 49
unique (*app.objects.secondclass.c_requirement.Requirement attribute*), 50
UNTRUSTED (*app.objects.c_operation.Operation.Reason attribute*), 53
update() (*app.utility.base_object.BaseObject method*), 61
update_agent_data() (*app.service.rest_svc.RestService method*), 60
update_chain_data() (*app.service.rest_svc.RestService method*), 60
update_config() (*app.service.rest_svc.RestService method*), 60
update_operation() (*app.objects.c_operation.Operation method*), 53
update_operation() (*app.service.rest_svc.RestService method*), 60
update_planner() (*app.service.rest_svc.RestService method*), 60
upload_file() (*app.api.rest_api.RestApi method*), 46
User (*app.utility.base_world.BaseWorld.Privileges attribute*), 63
username (*app.service.auth_svc.AuthService.User attribute*), 55

V

valid_config() (*app.contacts.contact_gist.Gist method*), 46
validate_login() (*app.api.rest_api.RestApi method*), 46
value (*app.service.data_svc.Adjustment attribute*), 56
Variation (*class in app.objects.secondclass.c_variation*), 50
Visibility (*class in app.objects.secondclass.c_visibility*), 50

W

wait_for_completion() (*app.objects.c_operation.Operation method*), 53
wait_for_links_completion() (*app.objects.c_operation.Operation method*), 53
walk_file_path() (*app.utility.base_world.BaseWorld static method*), 64
WebSocket (*class in app.contacts.contact_websocket*), 47
which_plugin() (*app.objects.c_ability.Ability method*), 51

which_plugin() (*app.objects.c_adversary.Adversary method*), 51
which_plugin() (*app.objects.c_planner.Planner method*), 53
write_result_file() (*app.service.file_svc.FileSvc method*), 58

X

xor_bytes() (*in module app.utility.payload_encoder*), 64
xor_file() (*in module app.utility.payload_encoder*), 64